

Q quantum electronics

Box 391262

Bramley
2018





Advanced Micro Devices

AmZ8000 User's Manual

The International Standard of Quality guarantees these electrical AQLs on all parameters over the operating temperature range: 0.1% on MOS RAMs & ROMs; 0.2% on Bipolar Logic & Interface; 0.3% on Linear, LSL. Logic & other memories.

© 1981 Advanced Micro Devices, Inc.

Advanced Micro Devices reserves the right to make changes in its products without notice in order to improve design or performance characteristics. The company assumes no responsibility for the use of any circuits described herein. Portions of the information contained in this manual were originated by Zilog, Inc. and are used here with their permission.

901 Thompson Place, P.O. Box 453, Sunnyvale, California 94086 (408) 732-2400 TWX: 910-339-9280 TELEX: 34-6306

Printed in U.S.A. 4/81 AMZ-291



Advanced Micro Devices

AmZ8000 User's Manual

The International Standard of Quality guarantees these electrical AQLs on all parameters over the operating remperature range: 0.1% on MOS RAMs & ROMs: 0.2% on Bipolar Logic & Interface; 0.3% on Linear LS & Logic & other memories on Linear LS.

@ 1981 Advanced Micro Devices, Inc.

Advanced Micro Devices reserves the right to make changes in its products withrout notice in order to improve design or performance characteristics. The company assumes no responsibility for the use of any circuits described herein. Portions of the information contained in this manual were originated by Zilog, Inc. and are used here with their permission.

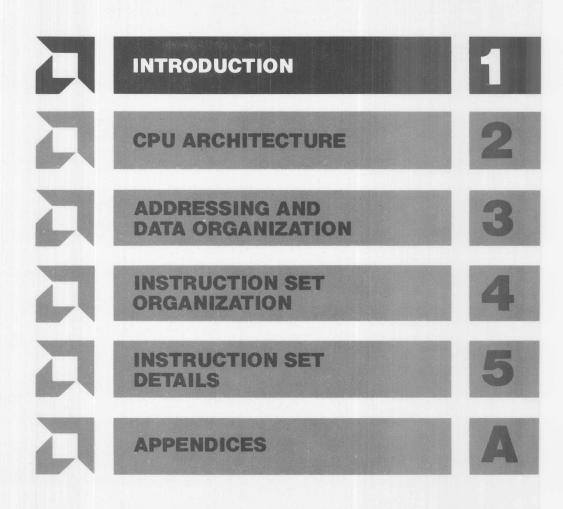
901 Thompson Place, P.O. Box 453, Sunnyvale, California 94086 (408) 732-2400 TWX: 910-939-9280 TELEX: 34-6306

Printed in U.S.A. 4/81 AMZ-231

CONTENTS

SECTION I	3.4	SEGMENTED ADDRESS FORMATS 3-3
INTRODUCTION1-1	3.5	DATA TYPES
1.1 HOW TO USE THIS BOOK1-1	3.6	ADDRESS MODES3-3
1.2 The AmZ8000 FAMILY1-1	3.6.1	Register Mode
1.3 AmZ8000 LITERATURE	3.6.2	Indirect Register3-4
OF STORY II	3.6.3	Direct Address
SECTION II	3.6.4	Immediate Mode3-5
CPU ARCHITECTURE2-1	3.6.5	Indexed Mode3-5
2.1 INTRODUCTION	3.6.6	Base Address Mode
2.2 CPU RESOURCES	3.6.7	Base Indexed Mode
2.2.1.1 Two CPU Versions	3.6.8	Relative Address
2.2.1.2 Register Resources	3.6.9	Autoincrement and Autodecrement3-5
2.2.1.3 Instruction Set Resources	3.6.10	Port Addressing Modes
2.2.1.4 Other Resources	0.0.10	Tort Addressing Wodes
2.2.1.5 Multimicroprocessor Support	SECTION	ON IV
2.2.2 Large Addressing Space		UCTION SET ORGANIZATION 4-1
2.2.3 Segmented Memory Addressing	4.1	INTRODUCTION
2.2.4 Memory Management	4.1	INSTRUCTION FORMAT
2.2.5 Code Density		
2.2.6 Throughput	4.3	INSTRUCTION DECODING
2.2.7 Compiler Efficiency	4.4	SEGMENTED AND NONSEGMENTED
2.2.8 Operating System Support		MODES
2.2.9 Software Support	4.5	CONDITION CODES 4-3
2.3 REGISTER ORGANIZATION2-6	4.6	INPUT/OUTPUT INSTRUCTIONS 4-3
2.3.1 General Registers	4.7	INSTRUCTION PREFETCH
2.3.2 Program Counter		(PIPELINING)4-4
2.3.3 Flag and Control Word (FCW) 2-8	4.8	EXTENDED INSTRUCTION
2.3.3.1 Flag Bits		PROCESSING
2.3.3.2 Control Bits		
2.5.5.2 CONTROL DITS	CECT	ON V
2.2.2.2. Looding a Now ECW	SECII	
2.3.3.3 Loading a New FCW		UCTION SET DETAILS 5-1
2.3.3.4 Processor Status Information 2-9		
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9	INSTR	UCTION SET DETAILS 5-1
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10	INSTR 5.1	UCTION SET DETAILS 5-1 INTRODUCTION 5-1
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10	INSTR 5.1	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND ENCODING 5-1
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10	5.1 5.2	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10	5.1 5.2 5.2.1	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10	5.1 5.2 5.2.1 5.2.2	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL DETAILS DETAILS 5-2 NOTATION KEY 5-7
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation 5-7
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III ADDRESSING AND DATA ORGANIZATION 3-1	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Source and Destination Notation 5-7
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III ADDRESSING AND DATA ORGANIZATION 3-1 3.1 INTRODUCTION 3-1	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Source and Destination Notation 5-7 Register Notation 5-7
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III ADDRESSING AND DATA ORGANIZATION 3-1 3.1 INTRODUCTION 3-1 3.2 ADDRESSING DATA 3-1	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4 5.6.5	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Register Notation 5-7 Operand Notation 5-7
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III ADDRESSING AND DATA ORGANIZATION 3-1 3.1 INTRODUCTION 3-1 3.2 ADDRESSING DATA 3-1 3.2.1 Addressing Data in Registers 3-1	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4 5.6.5 5.6.6	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Register Notation 5-7 Operand Notation 5-7 Address and Label Notation 5-7
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III ADDRESSING AND DATA ORGANIZATION 3-1 3.1 INTRODUCTION 3-1 3.2 ADDRESSING DATA 3-1 3.2.1 Addressing Data in Registers 3-1 3.2.2 Addressing Data in Memory 3-1	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4 5.6.5	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Source and Destination Notation 5-7 Operand Notation 5-7 Address and Label Notation 5-7 Condition Codes and
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III ADDRESSING AND DATA ORGANIZATION 3-1 3.1 INTRODUCTION 3-1 3.2.1 Addressing DATA 3-1 3.2.1 Addressing Data in Registers 3-1 3.2.2 Addressing Data in Memory 3-1 3.2.3 Data Storage in Memory 3-2	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4 5.6.5 5.6.6 5.6.7	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Source and Destination Notation 5-7 Register Notation 5-7 Operand Notation 5-7 Address and Label Notation 5-7 Condition Codes and Other Notations 5-8
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III ADDRESSING AND DATA ORGANIZATION 3-1 3.1 INTRODUCTION 3-1 3.2 ADDRESSING DATA 3-1 3.2.1 Addressing Data in Registers 3-1 3.2.2 Addressing Data in Memory 3-2 3.2.4 Data Contained in Instruction 3-2	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4 5.6.5 5.6.6 5.6.7	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Source and Destination Notation 5-7 Register Notation 5-7 Operand Notation 5-7 Address and Label Notation 5-7 Condition Codes and Other Notations 5-8 Special Character Notation 5-8
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III ADDRESSING AND DATA ORGANIZATION 3-1 3.1 INTRODUCTION 3-1 3.2 ADDRESSING DATA 3-1 3.2.1 Addressing Data in Registers 3-1 3.2.2 Addressing Data in Memory 3-2 3.2.4 Data Contained in Instruction 3-2 3.3 MEMORY ADDRESS FORMATS 3-2	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4 5.6.5 5.6.6 5.6.7	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Source and Destination Notation 5-7 Register Notation 5-7 Operand Notation 5-7 Address and Label Notation 5-7 Condition Codes and Other Notations 5-8 Special Character Notation 5-8 Instruction Page Key 5-10
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4 5.6.5 5.6.6 5.6.7	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Source and Destination Notation 5-7 Register Notation 5-7 Operand Notation 5-7 Address and Label Notation 5-7 Condition Codes and 0ther Notations 5-8 Special Character Notation 5-8 Instruction Page Key 5-10 INSTRUCTION PAGES 5-11
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception Processing 2-18 2.5 STATUS LINES 2-19 2.6 MEMORY AND I/O ADDRESSING 2-19 2.7 TIMING 2-20 SECTION III ADDRESSING AND DATA ORGANIZATION 3-1 3.1 INTRODUCTION 3-1 3.2 ADDRESSING DATA 3-1 3.2.1 Addressing Data in Registers 3-1 3.2.2 Addressing Data in Memory 3-1 3.2.3 Data Storage in Memory 3-2 3.2.4 Data Contained in Instruction 3-2 3.3.1 NS (Nonsegmen	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4 5.6.5 5.6.6 5.6.7	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Memonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Source and Destination Notation 5-7 Register Notation 5-7 Operand Notation 5-7 Address and Label Notation 5-7 Address and Label Notation 5-7 Condition Codes and 0ther Notations 5-8 Special Character Notation 5-8 Instruction Page Key 5-10 INSTRUCTION PAGES 5-11 EXTENDED PROCESSING </td
2.3.3.4 Processor Status Information 2-9 2.3.4 Stack Pointers 2-9 2.3.5 Refresh Counters 2-10 2.3.6 New Program Status Area Pointer 2-10 2.4 EXCEPTION PROCESSING 2-10 2.4.1 Types of Interrupts 2-10 2.4.2 Traps 2-10 2.4.3 New Program Status Area 2-11 2.4.4 Exception Processing Sequence 2-12 2.4.5 AmZ8001 and AmZ8002 Exception	5.1 5.2 5.2.1 5.2.2 5.2.3 5.2.4 5.3 5.4 5.5 5.6 5.6.1 5.6.2 5.6.3 5.6.4 5.6.5 5.6.6 5.6.7	UCTION SET DETAILS 5-1 INTRODUCTION 5-1 INSTRUCTION NOTATION AND 5-1 ENCODING 5-1 Instruction Mnemonics 5-1 Instruction Decoding 5-1 Address Mode Encoding 5-1 Use of Registers R0 and RR0 5-1 ASSEMBLER LANGUAGE SYNTAX 5-2 ASSEMBLER EXCEPTIONS 5-2 SUMMARY OF ARCHITECTURAL 5-2 DETAILS 5-2 NOTATION KEY 5-7 Appended Information 5-7 Addressing Mode and Segmentation Notation Notation 5-7 Source and Destination Notation 5-7 Register Notation 5-7 Operand Notation 5-7 Address and Label Notation 5-7 Condition Codes and 0ther Notations 5-8 Special Character Notation 5-8 Instruction Page Key 5-10 INSTRUCTION PAGES 5-11

		Listing by Opcode	
		dex	
ppendix E AmZ8000 Instruction Set	Oncode N	Map	E
		The state of the s	
		1189	
ppendix H Powers of 2 and 16		the the transfer of the transf	
Por Addressing Modes			
		Addressing Data in Memory	



INTRODUCTION

1.0 INTRODUCTION and a sebular energical and

1.1 HOW TO USE THIS BOOK

This book describes in detail the instruction set of the AmZ8001 and AmZ8002 CPUs. With both the system and application programmer in mind, software aspects of these devices are defined, including the basic architecture of the CPUs and hardware considerations important for a programmer's understanding. Additionally, assembler notations and other information is provided to support programming efforts. More complete details on device hardware or assembler operations can be found in other documents. (See Section 1.3.)

Advanced Micro Devices' School of Advanced Engineering offers courses on the AmZ8000 microprocessor family. Courses include a detailed introduction to the AmZ8000, assembly-level programming on the AmSYS8/8 using the 16-bit real-time emulator, and high-level language programming courses. Other courses cover the Am2900 bit-slice processor family, microprogramming with the AmSYS29, and related topics. Check with your AMD sales office for course outlines and schedules or call toll free (800) 538-8450 extension 3665.

1.2 THE AmZ8000 FAMILY

Advanced Micro Devices has undertaken a significant commitment to the world of 16-bit fixed-instruction-set processors. AMD is bringing to the market:

- · A new, advanced processor architecture
- · A complete family of LSI peripheral circuits
- · A complete family of system support circuits
- · A complete family of memories and memory support circuits
- Complete technical documentation
- Effective development system products
- Extensive support software

A large majority of future microprocessor applications will be serviced by a combination of single-chip microcomputer products such as the Am8048 series and by 16-bit microprocessors such as the AmZ8000. Where applications are simple enough, the 8-bit microcomputer chips will tend to be used. Increasing software costs and throughput requirements will cause the 16-bit CPUs to dominate the balance of the designs because they can answer these problems more efficiently. Conventional 8-bit microprocessors will serve a shrinking share of new designs.

In addition to significant increases in throughput that flow directly from the 16-bit structures, improved technology and more sophisticated architectures add even more performance.

Software cost savings are being realized through the use of more powerful instruction sets and sophisticated high-level languages such as PASCAL and C. Language compilers allow programmers to write, debug and document programs more quickly. And saving time is vitally important for such a labor-intensive activity, where costs are rapidly rising. The declining costs of technology-intensive LSI hardware can be used to improve software costs.

AmZ8000 processors, in terms of resources, system features, instructions, interface and architecture represent a major advance in microprocessor sophistication and system-level performance. The processors form the heart of a large family of components, systems, software, documentation and support. In addition to existing peripheral chips, a variety of new, advanced peripherals has been designed to support the AmZ8001 and AmZ8002 processors. Figure 1.2 shows these new MOS/LSI components as well as others that make up the AmZ8000 Family. Full information on these devices and the systems, software, documentation and support available is in The AmZ8000 Family Data Book. (See also Section 1.3.)

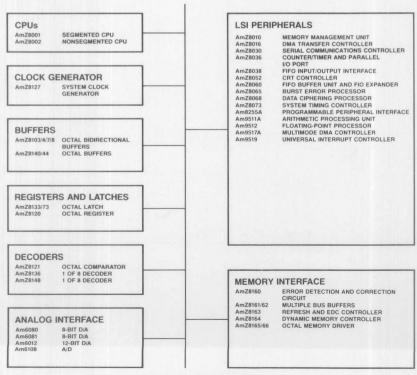


Figure 1.2 The AmZ8000 Family

An AmZ8000 Evaluation Board is available from Advanced Micro Computers for quick, hands-on experience with the AmZ8000. Called the Am96/4016, it is a complete small computer with RAM, ROM and several I/O ports. Available software includes a resident monitor and simple line-by-line assembler. The AmSYS™8/8 Microcomputer Development System supports the AmZ8000 Family as well as other microprocessors such as the Am8080A, Am8085A, Z80, and Am8048 Family. The system includes RAM, dual 8-inch floppy disk drives, several serial and parallel interfaces, and a cartridge disk.

Real-time emulation capabilities are provided by the RTE16 in both stand-alone and AmSYS8/8 plug-in versions. This allows real-time emulation, prototyping, and trace for AmZ8001, AmZ8002 and AmZ8010 MMU designs.

Powerful development software is available with the AmSYS8/8 to make the complex process of product development easier and

faster. The software includes a sophisticated disk operating system, macroassemblers, a linking loader, a powerful editor and debugger, and a PASCAL and C compiler with portable library. Additionally, an operating system for the AmZ8000 and cross-software products will be available.

Advanced Micro Devices was conceived on the premise that there is a place in the semiconductor community for a manufacturer dedicated to excellence. This attitude is manifested in many ways throughout the structure of the company and has been maintained throughout the life of AMD. In product assurance procedures, Advanced Micro Devices is unique. Only AMD processes all integrated circuits, commercial as well as military to the demanding requirements of MIL-STD-883. The AmZ8000 microprocessors and its family of support devices are no exception: every component is 100 percent screened to MIL-STD-883, Method 5004, Class C.

CPUS

AND STATES AND LATER AND LATER

1.3 AmZ8000 LITERATURE

The AmZ8000 Family Data Book

A compilation of data sheets and advanced information on the AmZ8000 CPUs, LSI peripherals, and system support, analog interface, dynamic memory system, and memory components. Also includes development support products: systems, evaluation boards, emulators and software. AmPUB-098

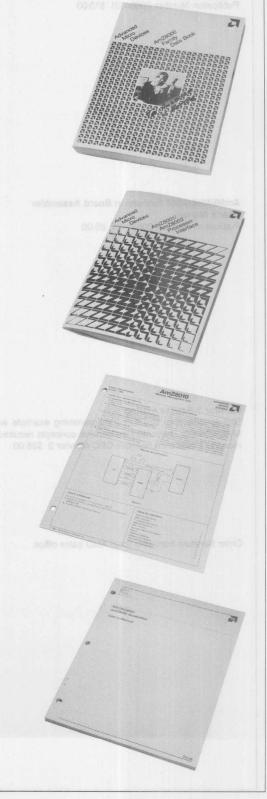
AmZ8001/AmZ8002 Processor Interface

Describes hardware interconnections between CPU and peripherals. Describes interrupt daisy chain and multimicroprocessor systems. AmPUB-089

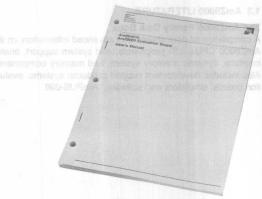
AmZ8010 Memory Management Unit

Complete product specification, including functional description, architecture, and timing. AMZ-192

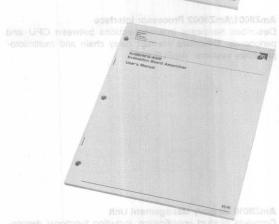
MACRO8000 Assembler User's Manual Publication Number 00680119 \$10.00



Am96/4016 AmZ8000 Evaluation Board User's Manual Publication Number 00680131 \$10.00



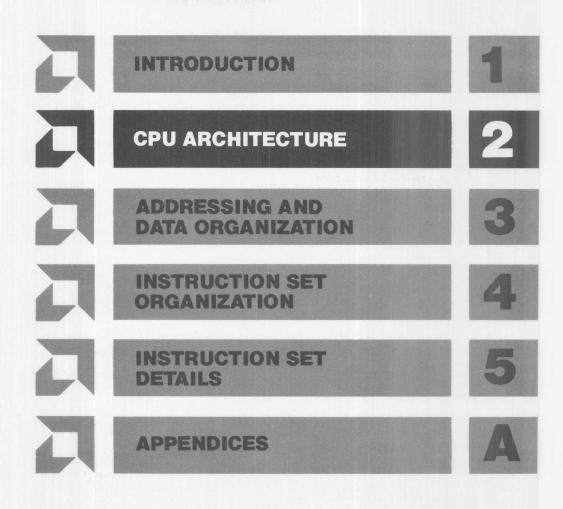
Am96/4016-ASM Evaluation Board Assembler User's Manual Publication Number 00680138 \$5.00



Monitor 3 Source Listing
Self-documenting structured programming example written in MACRO8000; demonstrates general concepts required in any monitor. Publication Number CEC-Monitor 3 \$25.00



Order literature from your local AMD sales office.



CPU ARCHITECTURE

2.0 CPU ARCHITECTURE

2.1 INTRODUCTION

The AmZ8000 is an advanced 16-bit microprocessor designed to span a wide variety of applications. Its features allow effective use in complex, high-throughput systems, yet it remains efficient for simpler systems, as well. This high-end 16-bit processor and its family of auxiliary devices support advanced systems ranging from simple stand-alone computers to complex parallel processing, multitasking and multiuser systems.

The AmZ8000 CPU represents a major advance in microcomputer architecture by offering many minicomputer and mainframe features in a microprocessor chip. The AmZ8000 CPU is available in two configurations, for use with or without an external memory management device that allows variable segment sizes and implements memory protection and relocation features. The segmented CPU, the AmZ8001, can directly address up to 8 megabytes of memory. The nonsegmented AmZ8002 can directly address 64K bytes.

Abundant CPU resources include numerous registers, many data element types, a large instruction set and eight user-selectable addressing modes. The CPU resources exhibit a consistency and regularity not found in previous microprocessor architectures. Regularity of register organization, of data types, instructions and addressing modes greatly simplifies the programming process and reduces program length.

The AmZ8000 CPU is partitioned into two modes for system (privileged) and normal (nonprivileged) operations. The instruction set similarly is partitioned and includes system-only instructions, such as the I/O instructions, not directly accessible to the normal user. Likewise, stack registers are duplicated in hardware to support both system and normal stacks. Such resources make advanced multiuser and multitasking systems very easy to implement.

The AmZ8000 achieves high throughput with a relatively low clock rate, and can use memories that have a comparatively long access time. The design also includes built-in memory refresh capability with a setable refresh rate that accommodates a variety of dynamic memories.

Compiler, compiler-produced and operating system code all run efficiently on the AmZ8000. The AmZ8000 supports compilers with features such as consistent instruction set, large address space, relocation, multiple stacks and specific instructions (PUSH, POP, INCREMENT and TEST).

Operating systems are supported by features such as system and normal modes, system and normal stack, specific instructions (SYSTEM CALL, LOAD PROGRAM STATUS and privileged instructions), and by a sophisticated interrupt and trap structure. This structure includes three types of interrupts (non-maskable, nonvectored and vectored) and four types of traps for system calls, privileged instructions, other special instructions, and segmentation.

Multimicroprocessor systems are supported in software by exclusion and synchronization instructions and in hardware by the Micro In input and Micro Out output.

2.2 CPU RESOURCES

Not only must the address space of an advanced architecture be large, but its CPU resources must be abundant enough for the solution of large problems.

The resources of the AmZ8000 CPU can be listed as follows:

- Regular general-purpose register architecture
 - useable as 8-bit, 16-bit, 32-bit, and 64-bit registers
 - user-defined as data, address, index, stack, counter registers
- 8M byte direct addressing range
 - separate code, data, stack spaces
 - 32M byte address space supported conveniently
- Software compatibility between the AmZ8002 and AmZ8001
- System (privileged) and normal (nonprivileged) operating partition
- Powerful instruction set with flexible addressing modes
 - over 110 instruction types
 - eight addressing modes
 - autoindexing instructions
 - string instructions with repeat and nonrepeat versions
- Data types including bits, digits, bytes, words, long words, byte strings, word strings, addresses
- Sophisticated exception processing capabilities
 - nonmaskable, nonvectored, and vectored interrupts
 - four types of traps, including segment trap from memory management unit
 - extended processing interface
- Five types of transfers including memory, I/O, and three daisy chains: interrupt request, bus request, and resource request
- Multimicroprocessing facilities

2.2.1.1 Two CPU Versions

The AmZ8000 CPU is offered in two versions: the AmZ8001 48-pin segmented CPU and the AmZ8002 40-pin nonsegmented CPU. They differ only in the manner and range of memory addressing. Physically, they are the same die with a metal pattern difference determining the CPU configuration.

The AmZ8001 can directly address 8M bytes of memory with its 23 bits of address. The AmZ8002 directly addresses 64K bytes of memory with 16 bits of address. These 16 bits are referred to as the offset address as opposed to the additional seven high-order address bits of the AmZ8001. These seven bits define the segment address and that is why the AmZ8001 is referred to as the segmented CPU. While the nonsegmented CPU directly addresses one segment of 64K bytes of memory, the segmented CPU address 128 such segments for a total of 8M bytes of memory.

In addition to the seven segment address pins the segment trap pin is available on the 48-pin AmZ8001. This provides an additional interrupt input to the CPU for interfacing to a memory management unit such as the AmZ8010.

In this book AmZ8000 CPU is used to refer to either the AmZ8001 or AmZ8002 CPU. Figures 2.2.1.1 and 2.2.1.2 show the functional and connection pin diagrams for the CPUs.

2.2.1.2 Register Resources

The AmZ8000 offers sixteen 16-bit general-purpose registers in addition to special system registers. All sixteen 16-bit registers may be used as accumulators and all but one can serve as index registers. The first eight of these 16-bit registers may be used as sixteen 8-bit byte registers. They may also be used as sixteen 16-bit word registers, as eight 32-bit long-word registers, or as four 64-bit quadruple-word registers.

The CPU architecture allows the creation and maintenance of stacks in memory. Any of the general-purpose registers (with one exception) can be designated as a stack pointer in the PUSH and

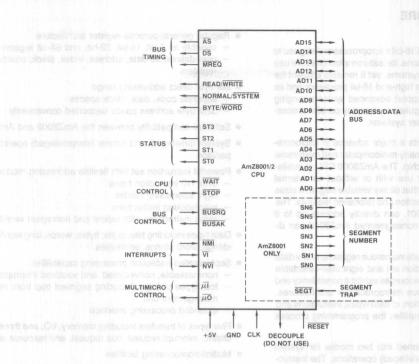
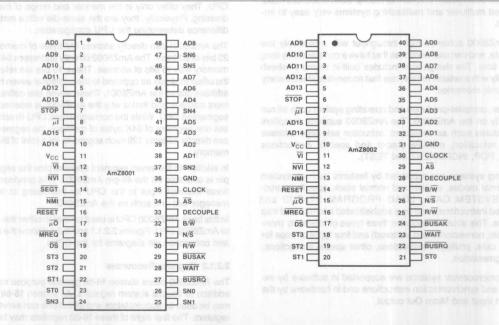


Figure 2.2.1.1 CPU Pin Functions



Note: Pin 1 is marked for orientation.

Figure 2.2.1.2 CPU Connection Diagram - Top View

POP instructions. For the CALL and RETURN instructions specific general-purpose registers are implied as stack pointers as described later.

To support the system and normal modes the specific generalpurpose registers implied as stack pointers are duplicated in hardware.

Special registers of the CPU include a program counter, a flag and control word register, a new program status area pointer register (for interrupt or trap context switching), and a refresh counter/register to facilitate dynamic memory system implementation.

2.2.1.3 Instruction Set Resources

The AmZ8000 provides the following groups of instructions:

- Load and Exchange
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Rotate and Shift
- Block Transfer and String Manipulation
- Input/Output
- CPU Control

These instructions are available for both the AmZ8002 nonsegmented CPU and the AmZ8001 segmented CPU.

Over 410 meaningful combinations of instruction types, data elements and addressing modes are available. The AmZ8000 also provides signed-multiply and signed-divide instructions implemented in hardware for both 16-and 32-bit values.

The AmZ8000 supports seven main data types: bits, BCD digits, bytes, words (16 bits), long words (32 bits), byte strings, and word strings. Additionally, many other data elements such as memory addresses, I/O addresses, segment table entries and program status words are also provided.

The eight user-selectable addressing modes include five main modes: Register(R), Indirect Register (IR), Direct Address (DA), Indexed (X) and Immediate (IM). For certain instructions, there are several other addressing modes: Base Address (BA), Base Indexed (BX), Relative Address (RA), Autoincrement and Autodecrement. String instructions such as I/O, translate, and others also have repeat and nonrepeat versions.

Compared to other microprocessors or to 16-bit minicomputers, the number and power of individual instructions has been greatly increased. Over 110 distinct instruction types are available with the AmZ8000 CPUs, compared to approximately 60 for the PDP 11/45. With few exceptions, byte, word and long-word data elements can be processed by all the instructions. Each instruction — again with few exceptions — can use any of the five main addressing modes.

Instruction prefetch (pipelining) has been designed into the CPU to improve overall execution times wherever possible. Also, special opcodes and hardware is designed into the CPU to allow expanding the instruction set externally with closely-coupled extended processing devices.

2.2.1.4 Other Resources

The AmZ8000 CPU outputs status information via four status lines (ST0-ST3) and the System/Normal line ($S\overline{\text{N}}$). These define the operating status of the CPU and can also be used to efficiently extend the addressing range. This is done by allocating physical memory to specific usage.

A very convenient division is to separate physical memory into code, data, system, and normal spaces for either an AmZ8001 or

AmZ8002 system. For example, the direct addressing range of 64K bytes of the AmZ8002 can be increased to 256K bytes with such a division.

Other resources and signals are provided for memory and I/O addressing and data transfers, interrupt and trap processing, CPU and bus control, and multimicro control. These are discussed elsewhere.

The AmZ8000 segmented CPU can be run in the nonsegmented mode. This is discussed elsewhere.

2.2.1.5 Multimicroprocessor Support

The AmZ8000 exclusion/serialization mechanism is designed for multimicroprocessor systems. Any CPU in a multimicroprocessor system can exclude all other asynchronous CPUs from any critical shared resource by using the Micro In $(\overline{\mu l})$ input and Micro Out $(\overline{\mu O})$ output in conjunction with the REQUEST, TEST μl , SET μO and RESET μO instructions.

In addition, the large address space of the AmZ8000 is a beneficial feature in multimicroprocessor systems.

2.2.2 Large Addressing Space

High-level languages, sophisticated operating systems, large data bases, large programs and decreasing memory prices are all accelerating the trend toward larger memories. The AmZ8001 can directly address up to eight megabytes of memory per address space. Four convenient separate address spaces exist in the AmZ8000: code and data for both the system mode and the normal mode.

Larger addressing spaces require longer addresses. This increases the size of instructions and requires larger registers (register pairs) when used for addresses. The impact of this is minimized in the AmZ8001 by segmented addressing features, the use of short addresses in many cases, and by the availability of a large number of general-purpose registers. Where the large address space is not required, the AmZ8002 CPU provides 64K bytes of space in each of four spaces and needs only a single 16-bit word for address reference.

AmZ8000 programs can directly access the entire address space. Since eight megabytes are directly addressable, each instruction has a full address mode where at least 23 bits are set aside for the address. However, the AmZ8000 also offers a mode in which the same address can be expressed by 16 bits in many frequent situations where the higher-order offset bits are zeros (short offset mode).

Alternative methods commonly employ fixed internal registers that contain address extensions. Although these methods use shorter addresses, the byte savings are lost because many instructions are required to explicitly manage the contents of the registers. The AmZ8000 can use these methods; however, it also provides direct addressing that removes the necessity for those extra instructions and unburdens the programmer from managing the register contents. It also has no speed loss for short offset addresses and the time lost in the case of long offsets is smaller than the time required to load an internal register.

Another important feature provided to the programmer is the ability to distinguish between system code and normal code, and — in both cases — the additional capability of distinguishing between instruction space, data space and stack space. If this feature is utilized, the AmZ8000 can address up to 48 megabytes of memory.

2.2.3 Segmented Memory Addressing

Segmentation is a powerful and useful technique because it forms an efficient way of dividing an address space into different functional areas. This allows a user to use a different segment for

each different area, such as areas for storing procedure instructions, holding global variables, storing multiple user common data or code, or serving as a buffer area for processing large, diskresident data bases.

The AmZ8001 uses segmented memory addressing not only to address the large amount of memory (8M bytes) but to provide the features required by advanced memory – intensive systems. For applications that do not require a large addressing space, the nonsegmented AmZ8002 CPU is available.

A segmented address is made of two parts: a segment number (7 address bits) and an offset value (16 address bits). The AmZ8001 can designate up to 128 segments each containing up to 64K bytes.

Code written for the nonsegmented AmZ8002 CPU can run in one segment of the segmented AmZ8001 CPU when it is operated in the nonsegmented mode. Thus, full code compatibility exists between the two versions.

2.2.4 Memory Management

Complex, large memory systems require not only capabilities of handling larger addressing spaces, but memory management features as well. Variable sized segments, logical to physical relocation, and memory protection can be provided with a separate memory management device. These features will extend the life of the architecture by avoiding memory address limitations that have hampered microprocessors in the past. A memory management system can be employed to provide system support such as segment swapping, memory access monitoring, and memory segment protection of various types (code only, DMA only, etc).

When segmentation is combined with an address translation mechanism to provide relocation capability, the advantages of segmentation are even greater. Then, segments can be of variable user-specified sizes and located anywhere in memory.

To meet such requirements of memory-intensive applications, a memory-management device has been designed to work closely with the AmZ8001 CPU. It is designated the AmZ8010 Memory Management Unit (MMU) and is based on the concept of segmented memory addressing.

The MMU manages the large address space by providing realtime segment relocation from logical to physical address space. It also monitors memory accesses, changes, and other protection attributes (size, CPU inhibit, read only, system only/normal inhibit, execute only/data-stack inhibit, CPU only/DMA inhibit).

This external memory management device essentially doubles the silicon area available to the microprocessor. Hence, it also doubles the logic available to the designer for implementing more features than otherwise would have been possible.

The AmZ8001 is designed to work with or without the MMU. If used, one or more MMUs can be employed for complex memory-intensive systems.

2.2.5 Code Density

Microprocessor speed is largely dependent on the number of executed instruction words. Therefore, code density is an important issue. The AmZ8000 offers several advantages in this respect.

The number of words required to specify frequent instructions (those instructions encountered by the assembler most frequently) has been minimized. This results in one word used for each JUMP RELATIVE, CALL RELATIVE, LOAD BYTE REGISTER IMMEDIATE and LOAD WORD REGISTER IMMEDIATE (for small immediate values).

A short offset mechanism is also designed to allow an address to be reduced to a single word. It can be automatically invoked by assemblers and compilers.

Finally, the largest reductions in size and increases in speed result from the consistent and regular structure of the architecture, and the greater power of the instruction set — factors that allow fewer instructions to accomplish a given task. Compared to previous microprocessor designs, AmZ8000 architecture is more regular because its registers, address modes and data element types can be used in a more orderly fashion. Any general-purpose register can be specified to be an accumulator, index register or source register. With few exceptions, all addressing modes can be used with all instructions. Similarly, all the various data types (again with few exceptions) can be used with all the addressing modes.

2.2.6 Throughput

Meaningful evaluations of computer performance will include comparisons based on the execution times of typical programs in typical applications. For the AmZ8000, these applications normally involve high-level language compilers, operating systems and large data-base management. In such areas, the AmZ8000 is five to ten times faster than existing 8-bit microprocessors and two to five times faster than modern 16-bit microprocessors or popular minicomputers such as the PDP 11/34. Furthermore, the AmZ8000 does this with proven N-channel MOS technology and a moderate 4MHz clock rate that allows the use of lower-cost dynamic RAMs. The AmZ8000 is also available in a 6MHz version. The AmZ8000 overlaps instruction execution with next instruction fetch and avoids the problems associated with deep unconditional prefetching.

The AmZ8000 can achieve this high degree of performance because its regular architecture does not have critical bottlenecks and because the sophisticated instruction set substantially reduces the number of executed instructions. Some examples of this sophistication are:

- 32-bit operations (including Multiply and Divide) in single instructions
- String manipulation, including Compare and Translate
- Block I/O instructions
- Direct addressing of the entire memory
- Two operating modes (System/Normal or Supervisor/User)
- Powerful interrupt handling

The combination of the powerful instruction set and regular architecture reduces the number of instructions required to execute a program and boosts AmZ8000 performance into the range of well-established minicomputers. (The AmZ8000 is faster than the PDP 11/45 and only slightly slower than the PDP 11/70.)

The following comparisons demonstrate the high execution speed of the AmZ8000 and compare it to the PDP 11/45. Only the 4MHz device is shown in the comparison; times for the 6MHz version would be proportionally shorter. The AmZ8000 is faster in all except Multiply. These tables and graphs show that the advantage of the AmZ8000 is more pronounced when the comparisons involve greater instruction power, more sophisticated addressing modes or longer word lengths. Actual applications programs will take advantage of the more sophisticated aspects of the instruction set and architecture, and will therefore run considerably faster on the AmZ8000 than the PDP 11/45.

TABLE 2.2.6.1 EXECUTION TIME FOR LDB R, src (μs)

Source Addressing Mode	AmZ8000 at 4MHz	PDP 11/45 with 8K	
Register	0.75	0.90	
Indirect Register	1.75	1.88	
Direct Addressing	2.25	2.78	
Indexed Addressing	2.50	2.78	
Immediate bestlern od	1.00	1.88	

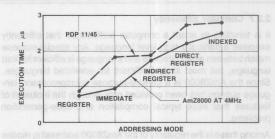
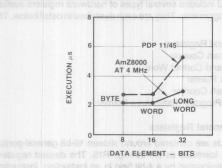
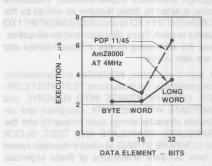


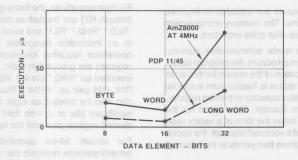
Figure 2.2.6.1 Execution Times LDB R, src for Various Addressing Modes



a) Execution Times for LD R, DA



b) Execution Times for ADD R, DA



ETH STREET OF DO DE LOS TADRES QUOD VE C) Execution Times for MULT R, DA

PDP is a registered Trademark of Digital Equipment Corporation Figure 2.2.6.2

TABLE 2.2.6.2 EXECUTION TIMES FOR LD, ADD AND MULT

	brow-pnol a eniatra	AmZ8000 at 4HMz				PDP 11/45 With 8K		
Operation	Data Type	Inst.	Bytes	Cycles	μs	Inst.	Bytes	μs
exocification for Ed. J.	Byte	the sales	4	2.9	2.25	or of the star	omem 4 (Inel)	2.78
LD R, DA	Word	1 - 1	4	1091000	2.25	angest under a	to sufer 4 rango	2.78
	Long Word	seec 1 no to	4	12	3.00	Ten 2 TAT	MAROBAN D	5.56
If to hataigen tast	Byte	i farifi prifi si i	uP ,elc4nsva	9	2.25	2 2 115	hines hife moisu	3.68
ADD R, DA	Word Word	RA is fore fin	,000F4 BUD	9	2.25	struction that	TES 4 TE	2.78
	Long Word	1	.no 64 bes	15	3.75	3	lecord 10 la redo	6.46
mode), a registr	Byte	151503 100	Ama 8 mai	87	21.75	2	6	6.61
MULT, DA	Word	s of Hapsey	4 109	70	17.50	1	4	5.56
	Long Word	at contract is	90004	350	88	17	42	33.94*

^{*}If double floating point is used it is one instruction four bytes and 7.23 µs

2.2.7 Compiler Efficiency

It is tempting to adapt a computer architecture that efficiently executes a particular high-level language. Any special-purpose match between an architecture and a language is efficient for that language, but most likely inefficient for unrelated languages. Since the AmZ8000 is a general-purpose microprocessor, language support has been provided only through the inclusion of features that ease typical compilation and code-generation problems.

Among these is the regularity of the AmZ8000 addressing modes and data element types. Note that any register (except R0) can be used as a stack pointer by its sophisticated PUSH and POP instructions. Segmentation and relocation are useful features for high-level language procedure implementation. Procedure parameter passing is aided by these features as well as by the instructions INCREMENT BY 1 TO 16 and DECREMENT BY 1 TO 16, which are useful in stack frame allocation and de-allocation. For stack frames, the addressing modes of Base Address and Base Indexed are also useful.

Testing of data, logical evaluation, initialization and comparison of data are made possible by the instructions TEST, TEST CONDITION CODES, LOAD IMMEDIATE INTO MEMORY, and COMPARE IMMEDIATE WITH MEMORY. Compilers and assemblers manipulate character strings quite frequently and the instructions TRANSLATE, TRANSLATE AND TEST, BLOCK COMPARE, and COMPARE STRING all result in dramatic speed improvements over software simulations of these important tasks.

2.2.8 Operating System Support

Interrupt and task-switching features are included to improve operating system implementations. The memory-management and compiler-support features are also quite important.

The interrupt structure has three levels: nonmaskable, nonvectored and vectored. When an interrupt occurs, the program status is saved on the stack with an indication of the reason for this state switching before a new program status is loaded from a special area of memory. The program status consists of the flag register, the control bits and the program counter. The reason for the occurrence (saved on the stack) takes the form of a vector read from the system bus. In the case of a vectored interrupt, the vector also determines a jump table address that points to the interrupt processing routine.

The inclusion of system and normal modes improves operating system organization. In the system mode, all operations are allowed; in the normal mode, certain system instructions are prohibited. The SYSTEM CALL instruction allows a controlled switch of mode, and the implementation of traps enforces these restrictions.

Traps result in the same type of program status saving as interrupts: in both cases, the information saved is pushed on a system stack that keeps the normal stack undisturbed. The LOAD MULTIPLE REGISTER instruction allows the contents of registers to be saved efficiently in memory or on the stack. Running programs can cause program status changes under direct software control with the LOAD PROGRAM STATUS instruction.

Finally, exclusion and serialization can be achieved with the "atomic" TEST AND SET instruction that synchronizes asynchronous cooperating processes.

2.2.9 Software Support

AMD offers a full range of applications support and system support software for the AmZ8000 family. Language compilers, assemblers, utilities, translators for existing programs, etc., are all

available to AMD customers. Present users of Z80, Am8080A and Am8085A microprocessors can easily convert their existing software into AmZ8000 code.

Even though there are planned similarities to previous processors, the AmZ8000 represents a major architectural advance over existing designs. There are more instructions, more data types, more addressing modes, larger addressing spaces, and greater instruction complexity. Thus, the full benefits of the AmZ8000 architecture will only be realized by making code changes to existing programs to take advantage of the greater available performance.

2.3 REGISTER ORGANIZATION

The CPU includes several types of hardware registers available to the programmer. They are each described in detail below. They are:

- 1. General Registers
- 2. Program Counter
- 3. Flag and Control Word
- 4. Stack Pointer
- 5. Refresh Counter
- 6. New Program Status Area Pointer

2.3.1 General Registers

The CPUs are centered around sixteen 16-bit general-purpose registers identified as R0 through R15. The desired register is usually designated by a 4-bit field in an instruction. Instructions operate on bit, byte (8-bit), word (16-bit), long word (32-bit) or quad (64-bit) operands. Refer to Figures 2.3.1.1 and 2.3.1.2.

For byte operations, the first eight general-purpose registers (R0 through R7) are treated as sixteen 8-bit registers identified as "RL0," "RH0," "RL1" and so on to "RL7" and "RH7." A 4-bit field in an instruction designates the desired byte register. For operations requiring long-words, the 16-bit general-purpose registers are grouped in pairs. For example, the R0, R1 pair is identified as "RR0," the R2, R3 pair as "RR2" and so on to the R14, R15 pair as "RR14." Thus, the general-purpose registers can also be treated as eight 32-bit registers. The three most significant bits of a 4-bit field in an instruction designate the desired register pair and the fourth bit should be zero.

For certain 64-bit operands (multiply and divide), the general-purpose registers can also be grouped into quads. For example, the R0, R1, R2 and R3 group is identified as "RQ0," the R4, R5, R6 and R7 group as "RQ4" and so on to the R12, R13, R14 and R15 group as "RQ12." The two most significant bits of a 4-bit field in an instruction designate the desired quad register and the remaining two bits should be zero. Table 2.3.1 depicts this organization and gives the four-bit designation used in the source and destination fields of instructions.

The registers may contain operands or address information. When a register pair contains a long-word operand, the even numbered register of the pair holds the most significant 16-bit data while the odd numbered register of the pair holds the least significant 16-bit data. When a register quad is specified for 64-bit data, the first register holds the most significant 16-bits and the last register of the quad holds the least significant 16 bits. For example, R0 is the first register and R3 is the last register of the quad RQ0, R4 is the first and R7 is the last of the quad RQ4 and so on.

In the AmZ8001 (operating in the segmented mode), a register pair will be needed to specify the required 23-bit address. The 7-bit segment number is always specified in the even numbered register and 16-bit offset is specified in the odd numbered register of the pair.

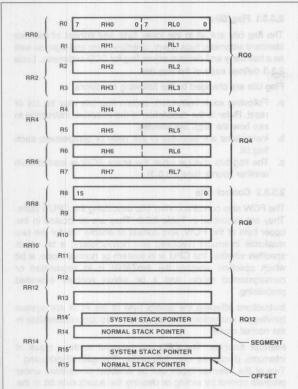


Figure 2.3.1.1 AmZ8001 General Registers

TABLE 2.3.1 GENERAL REGISTER ORGANIZATION AND DESIGNATORS

Register Designator	Byte Mode	Word Mode	Long Word Mode	Quadruple Word Mode
0000	RH0	R0	RR0	RQ0
0001	RH1	R1	-	-
0010	RH2	R2	RR2	-
0011	RH3	R3	-	-
0100	RH4	R4	RR4	RQ4
0101	RH5	R5	B	chiso old _
0110	RH6	R6	RR6	
0111	RH7	R7	-	- 1
1000	RL0	R8	RR8	RQ8
1001	RL1	R9	desirantine t	- Flasuil o
1010	RL2	R10	RR10	o flusteri -
1011	RL3	R11	- (6	- (MSB -
1100	RL4	R12	RR12	RQ12
1101	RL5	R13	uro regnia di	lessi ts -
1110	RL6	R14	RR14	
1111	RL7	R15	-	(183 ON _

(-Reserved)

All general purpose registers can be used as accumulators. However, R0 in the AmZ8002 (and RR0 in the AmZ8001) cannot be used as an index register or memory pointer. Refer to the section on Address Modes (3.6) and Section 5.2.4.

The highest order general-purpose registers are used as implied stack pointers. For a description of this refer to the section entitled Stack Pointers (2.3.4).

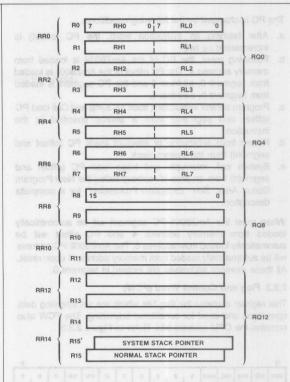


Figure 2.3.1.2 AmZ8002 General Registers

2.3.2 Program Counter

Refer to Figure 2.3.2

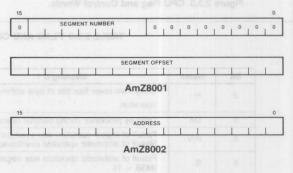


Figure 2.3.2 CPU Program Counters

The program counter points to the address of the next instruction to be fetched from memory. It is a 16-bit register in the AmZ8002 and a register pair in the AmZ8001. The pair contains a 7-bit segment number and a 16-bit offset. The AmZ8000 addresses bytes in memory, instructions occupy full words and are located on even byte addresses (the LSB of the PC is always 0). Following the fetch of each word of an instruction from memory, the PC is incremented by two. In the AmZ8001, only the offset is incremented; the segment number is not changed by incrementing the PC offset.

The PC is changed by the following conditions:

- After fetching an instruction word, the PC (offset) is incremented by two.
- b. Following reset, the PC of the AmZ8002 is loaded from memory address 4. The PC offset of the AmZ8001 is loaded from segment 0, address 6 and the PC segment is loaded from segment 0, address 4.
- Program control instructions such as Jump and Call load PC (offset and segment) from a source specified in the instruction.
- Return from subroutine or interrupt loads PC (offset and segment) from the system stack.
- e. System call, interrupts and traps load PC (offset and segment) from an area of memory called the New Program Status Area. See "Exception Processing" for a complete description.

When reset the AmZ8001 PC segment will be automatically loaded from memory address 4 and PC offset will be automatically loaded from address 6. The AmZ8002 PC address will be automatically loaded from memory address 4 upon reset. All these memory addresses are located in segment 0.

2.3.3 Flag and Control Word (FCW)

This register contains the flag bits which are set following data operations and used for conditional branching. The FCW also contains the CPU controls bits. Refer to Figure 2.3.3.

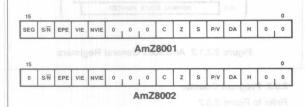


Figure 2.3.3 CPU Flag and Control Words

2.3.3.1 Flag Bits

The flag bits are all in the lower byte and consist of the four standard arithmetic flags (carry, overflow, zero and sign) as well as a half carry and decimal adjust flag for BCD arithmetic. Table 2.3.3 defines each of the flag bits.

Flag bits are changed by the following conditions:

- Following each instruction, certain flag bits may be set or reset. Refer to the description of the individual instruction to see how the flags are affected.
- Instructions are available to set, reset or complement each flag bit.
- The flag bits change when the entire FCW is loaded from another source (see 2.3.3.3).

2.3.3.2 Control Bits

The FCW also contains control bits describing the CPU's state. They are described in Table 2.3.3. They are all located in the upper byte of the FCW, and consist of enable bits for the two maskable interrupts (vectored and nonvectored); a bit which specifies whether the CPU is in system or normal mode, a bit which specifies whether the AmZ8001 is in segmented or nonsegmented mode, and a bit which enables extended processing.

Instructions affecting the control byte of the FCW are system (privileged) instructions making these control bits inaccessible in the normal mode.

Interrupt Enables: The CPU can respond to three types of interrupts, discussed more fully under "Exception Processing." Two of the interrupt types can be enabled or disabled under program control by setting or clearing the appropriate bit in the FCW. This can be done by individual instructions or by loading a new FCW from another source (see 2.3.3.3).

Extended Processing Enable: The CPU can allow close-coupled extended processing units with the enabling of this control bit. If not set, then upon the execution of any of the special (extended processing) opcodes, a trap will result. See "Extended Processing" for a description of this.

TABLE 2.3.3 FLAG AND CONTROL WORD BITS (FCW)

Bit	Name	Meaning if 1	Meaning if 0
2	Н	Carry from lower four bits of byte arithmetic operation.	No carry.
3	DA	Used by processor during decimal operation.	H BHZ HZ
4	P/V	Result of logical operation was even parity. Result of arithmetic operation overflowed.	Result of logical operation was odd parity. Result of arithmetic operation did not overflow.
5	S	Result of arithmetic operation was negative (MSB = 1).	Result of arithmetic operation was zero or positive (MSB = 0).
6	Z	Result of arithmetic or logical operation was all zeroes.	Result of arithmetic or logical operation contains at least a single one.
7	С	Carry occurred from MSB (sign) bit of arithmetic operation.	No carry.
11	NVIE	Nonvectored interrupts enabled.	Nonvectored interrupts disabled.
12	VIE	Vectored interrupts enabled.	Vectored interrupts disabled.
13	EPE	Enable extended processing.	Implements trap if special extended processing opcode is executed.
14	S/N	Processor in system mode.	Processor in normal mode.
15	SEG	AmZ8001 operating in segmented mode.	AmZ8001 emulating AmZ8002 (nonsegmented).

System/Normal Mode: The CPU can operate in either of two modes; called "system" and "normal" (or "user"). The mode is specified by a bit in the FCW. The differences between the two modes are:

- a. Certain instructions cannot be executed in normal mode. These include those instructions which are generally used by an operating system. They are:
 - All I/O instructions
 - Instructions affecting the control bits (upper byte of FCW)
 - Instructions affecting the multiprocessor facility

An attempt to execute one of these instructions results in a trap called "privileged instruction trap." See "Exception Processing" for a detailed description of what happens.

b. A different register is used for R15 (R14 and R15 in the AmZ8001). This general register is intended for use as the primary stack pointer. While any general register except R0 can be used as a stack pointer, certain instructions automatically invoke R15 as the stack pointer for saving system status. The CPU uses a different hardware register (or register pair) for R15 when in the system mode than in the normal mode as registers are duplicated in hardware for both system and normal stack pointer.

The extra register is called R15' to distinguish it from the normal mode R15. In the AmZ8002, only R15 is separated. In the AmZ8001, both R14 and R15 are separated, since the pair is needed to hold both segment and offset of the stack address. A reference to R15 (or R14) when the processor is in system mode actually refers to R15' (or R14'). The same reference in normal mode refers to the normal R15 (or R14). All other registers are the same in either mode.

The System/Normal bit is changed only by loading a new FCW (see 2.3.3.3).

Nonsegmented Mode of the AmZ8001. In the AmZ8002, addresses are completely contained in a single 16-bit word. The AmZ8001 uses a segment and offset, requiring two words. All the instructions are identical; only the address references are different. The AmZ8001 can be switched to a nonsegmented mode to execute software assembled for the AmZ8002. This switch is accomplished by the segment bit (15) in the flag and control word.

The segment bit is available only on the AmZ8001. It is always zero on the AmZ8002. It can be changed on the AmZ8001 only by loading a new FCW (see 2.3.3.3). Refer to "Segmented and Nonsegmented Modes" for a description of this feature.

2.3.3.3 Loading a New FCW

The entire Flag and Control Word can be loaded in the following ways:

- a. An instruction is available to load it from any source.
- b. On an interrupt or trap it is loaded automatically from the New Program Status Area (see "Exception Processing").
- c. On a return from an interrupt or trap it is restored to its previous state by a reloading from the system stack.
- d. Following RESET, the FCW is loaded from memory location 2 (segment 0 of AmZ8001).

2.3.3.4 Processor Status Information

The contents of the program counter and flag and control word are collectively called the Processor Status Information. When an interrupt or trap occurs, current processor status information is saved and new processor status information is loaded from the New Program Status Area of Memory (see "Exception Processing").

Figure 2.3.3.4 illustrates how the program status groups of the AmZ8001 and AmZ8002 differ. In the nonsegmented CPU the program status group consists of two words: the PC and FCW. In the segmented CPU the program status group consists of four words: a two-word PC, the FCW, and an unused word reserved for future use

With the exception of the segment enable bit in the AmZ8001 program status group, the flags and control bits are the same for both CPUs.

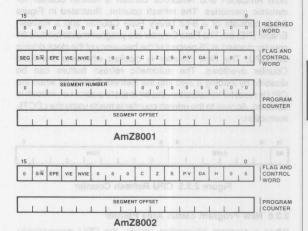


Figure 2.3.3.4 CPU Processor Status Information

2.3.4 Stack Pointers

The architecture allows the creation and maintenace of stacks in the memory. Any of the general-purpose registers (except RR0 in AmZ8001 and R0 in AmZ8002) can be designated as stack pointers in the PUSH and POP instructions. However, for the CALL and RETURN instructions, specific general-purpose registers are implied as stack pointers.

In the AmZ8001 the general-purpose register pair RR14 is the implied stack pointer. The 7-bit segment number is contained in R14 and the 16-bit offset value is contained in R15. The segment number and the offset value form a 23-bit segmented address. For the AmZ8002 the general-purpose register R15 is the implied stack pointer and contains the required 16-bit address. It should be remembered that the implied stack pointers are still general-purpose registers. In other words, certain implied general-purpose registers are given stack pointer attributes in addition to their normal general-purpose characteristics.

The processors can operate in one of two selectable modes: System and Normal. The System mode is sometimes called "supervisor" or "privileged" mode and the Normal mode is sometimes known as "program" or "nonprivileged" mode. Separation of system and normal stacks is desirable in order to facilitate sophisticated system designs. This is accomplished by providing System Stack Pointer in addition to Normal Stack Pointer by duplicating these registers in hardware.

In the AmZ8001 two additional registers, R14' and R15', are provided corresponding to R14 and R15. When the AmZ8001 is operating in the system mode, R14' will be used as the general-purpose register whenever R14 is specified. Similarly R15' will be used instead of R15 in the system mode for both AmZ8001 and AmZ8002. Thus, the register pair R14', R15' (identified as RR14') is the implied System Stack Pointer for the AmZ8001 and R15' is the implied System Stack Pointer for the

AmZ8002. Although R14 and R15 are not used in the system mode, instructions are provided such that these two general-purpose registers can be accessed without actually switching the operating mode. The System Stack Pointer will be used during program interruptions to save the pre-interrupt status irrespective of the selected operating mode. Refer to Figures 2.3.1.1 and 2.3.1.2 for a look at these.

2.3.5 Refresh Counter

Both AmZ8001 and AmZ8002 contain a refresh counter for dynamic memories. The refresh counter, illustrated in Figure 2.3.5, consists of a 9-bit binary Rate Counter and a Refresh Enable (RE) bit. The Rate Counter is a programmable modulo 64 counter clocked at 25 percent of the frequency of the clock driving the CPU. The Row Counter is clocked whenever the Rate Counter overflows. The automatic refresh feature can be disabled by loading a zero into the Refresh Enable bit when the CPU is reset for initialization, this bit is set to a zero, i.e., refresh is disabled. Access to the refresh counter is made using the LDCTL instruction.

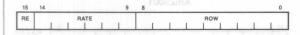


Figure 2.3.5 CPU Refresh Counter

2.3.6 New Program Status Area Pointer

When a program interruption occurs, the CPU automatically saves the program status in the system stack. New program status will be loaded into the FCW and PC. This new program status is obtained from a location in the memory called New Program Status Area. The user defines the begining location of this area by setting the New Program Status Area Pointer(NPSAP). The NPSAP may point to a 256-byte boundary anywhere into memory. The NPSAP is shown in Figure 2.3.6. In the AmZ8001 it consists of two 8-bit registers, one for the segment and one for the most significant eight bits of the offset. On the other hand, only one 8-bit register is used in the AmZ8002. This register specifies the most significant 8-bits of the 16-bit address. The NPSAP is changed by using the LDCTL instruction. See "Exception Processing" for details of the New Program Status Area.

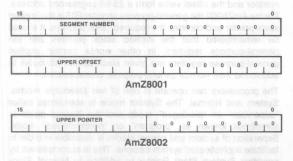


Figure 2.3.6 New Program Status Area Pointer

2.4 EXCEPTION PROCESSING

Program interruptions are divided into two groups – interrupts and traps. In general, an interrupt is an external asynchronous event needing the CPU's attention. A trap usually is a synchronous event resulting from the execution of certain instructions under some specified condition. Also an interrupt

(except for the nonmaskable type) may be disabled in the CPU by an appropriate control bit in the FCW; traps cannot be disabled. Procedures followed by the CPU are essentially the same for interrupts and traps.

The descending order of priority for traps and interrupts is: internal traps, nonmaskable interrupt, segmentation trap, vectored interrupt, and nonvectored interrupt. These types are defined below.

When an interrupt or trap occurs, the CPU automatically switches to the system mode and saves the program status information (PC and FCW) plus identifier word on the system stack. The identifier supplies the reason for the interruption and is returned by the interrupt or trap. For external traps (segmentation error) and interrupts, the identifier is the vector on the data bus read by the CPU during the interrupt-acknowledge or trap-acknowledge cycle. Refer also to Figure 5.5.6.

2.4.1 Type of Interrupts

There are three types of interrupts serviced by the CPU. In order of decreasing priority they are: nonmaskable, vectored and nonvectored

The nonmaskable interrupt is always active; it cannot be disabled under program control. It is typically used for power-failure sensing. The nonmaskable interrupt results in a jump to a user defined location in memory. The interrupting device may pass a 16-bit identifier containing information about the interrupt to the program.

The vectored interrupt may be masked by clearing bit 12 in the FCW register. If bit 12 is set, then an interrupt causes a jump in memory to an address defined indirectly by the vector supplied by the interrupting device. The device generating the interrupt may pass an 8-bit identifier to the program, along with the 8-bit vector.

The nonvectored interrupt may be masked by clearing bit 11 in the FCW register. If bit 11 set, then an interrupt causes a jump in memory to an address defined by the user. A nonvectored interrupt always results in a branch to the same location. The interrupting device may pass a 16-bit identifier to the program.

2.4.2 Traps of the land the land the land to the land

There are four types of traps in the CPU: system call instruction, special opcode, privileged instruction in normal mode, and segmentation error. They cannot be disabled.

The identifier in the case of traps (except segmentation error) is the first word of the instruction that caused the trap. This word always contains the instruction opcode.

System Call: The system call instruction is a software trap. This instruction contains an 8-bit field which can be used by the programmer to pass information to the system. The system call results in a branch to a memory address specified by the user.

Special Opcode: There are six special opcodes in the AmZ8001 and AmZ8002 which allow for extended processing or results in a special opcode trap (see "Extended Processing"). An attempt to execute one of them (if the EPE control bit is not set) results in this trap occurring. The program branches to a memory address specified by the user. The opcode causing the trap is passed to the program as an identifier. The six special opcodes, in hex, are: OExx, OFxx, 4Exx, 4Fxx, 8Exx, 8Fxx.

Privileged Instruction: This trap results when an attempt is made to execute one of the system-only instructions when the CPU is in the normal mode. The program branches to a user-defined address and the offending opcode is passed as an identifier. The privileged instructions are: all I/O instructions, instructions which inspect or modify the control bits of the FCW, and those that are involved in the multimicro (multiprocessor) facility.

Segmentation Error: This trap is used by the AmZ8001 to respond to an addressing error, generally an offset which is outside the defined boundary for a segment or some other segment addressing violation detected by the Memory Management Unit (AmZ8010). The identifier is supplied by the AmZ8010 and is discussed fully in the MMU documentation.

2.4.3 New Program Status Area

After the old program status information (PC and FCW) is stored in the system stack following an interrupt or trap, the new program status information is automatically loaded into the PC and FCW from a specified area in memory. This area is called the New Program Status Area (NPSA) and it contains information for context switches due to each type of interrupt and trap.

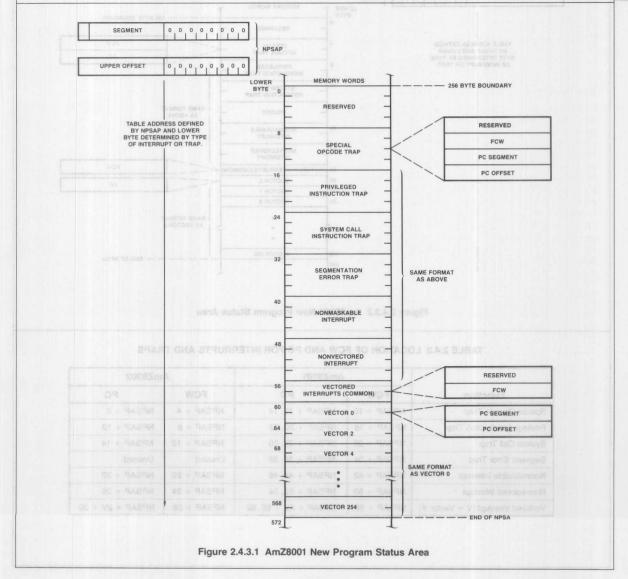
The CPU allows the location of the NPSA to be anywhere in the addressable memory space, although it must be aligned to a 256 byte boundary. The New Program Status Area Pointer (NPSAP) register specifies the begining address of the NPSA (see 2.3.6).

The CPU uses this address along with an offset, depending upon which type of interrupt or trap occurs, to index into the NPSA table. Table 2.4.3 defines the location of the data in the NPSA

The NPSAP is a one-word register in the AmZ8002, the lowest byte of which is zero. In the AmZ8001 the NPSAP is stored in a two-word register; the first contains the segment number and the second contains the offset. Again, the lowest byte is zero. The NPSAP is loaded and read using the LDCTL instruction.

(The NPSA must be located in code memory space if separate memory spaces for code and data are being used or when using the code attribute with the Memory Management Unit. This is because of the Status Line decode for a code reference; refer to the MMU documentation.)

The NPSA contains a table of the new flag and control words and starting address (PCs) of service routines for each type of interrupt and trap. The table format is shown in Figures 2.4.3.1 and 2.4.3.2 for the AmZ8001 and AmZ8002, respectively.



For the AmZ8001 consecutive locations contain 1) reserve word, 2) new FCW, 3) new PC segment and 4) new PC offset for all traps and interrupts except vectored interrupts. The vectored interrupts share a common reserved word and FCW, and each vector has its own PC segment and offset. For the AmZ8002 there are just two words for each entry, the FCW and PC. The vectored interrupts share a common FCW and have separate PCs. Refer to Section 2.4.5.

2.4.4 Exception Processing Sequence

When an interrupt or trap occurs, the current program status information is pushed onto the system stack in the following order: program counter (or PC offset, then segment in the AmZ8001), contents of FCW register, Identifier. This is illustrated in Figure 2.4.4.1. (See also Figure 5.5.6.) The identifier is the data

defined above for the various types of interrupts and traps. The service routine can access the identifier by simply addressing the word pointed to by the system stack pointer since it is at the top of the stack

The program counter and flag control word are reloaded with values located in memory in the New Program Status Area. The new program status will be loaded from an address whose upper 8 bits (and segment number in the AmZ8001) are stored in the NPSAP and whose lower 8 bits are determined by the particular trap, interrupt or vector occurring. The CPU, of course, then begins execution immediately at the address found in the New Program Status Area. This address can also be anywhere in memory. The entire process is illustrated in Figure 2.4.4.2.

To service an interrupt or trap requires 34 CPU clocks.

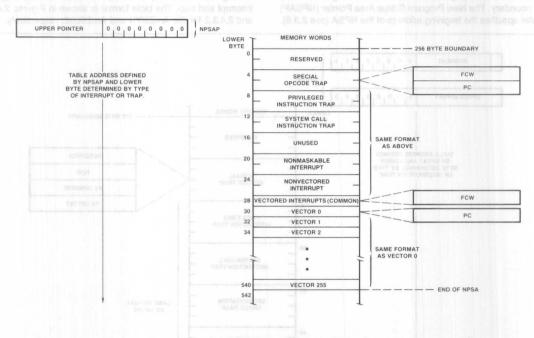
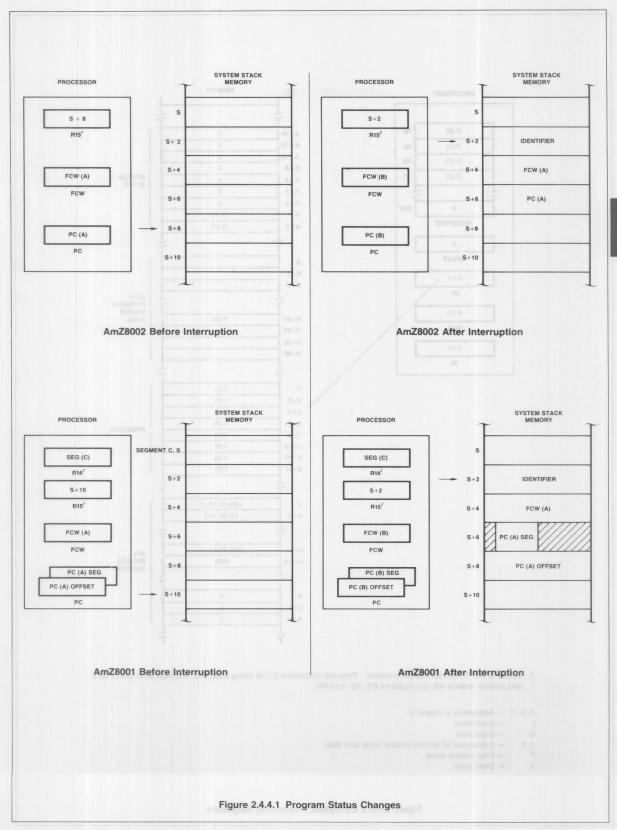
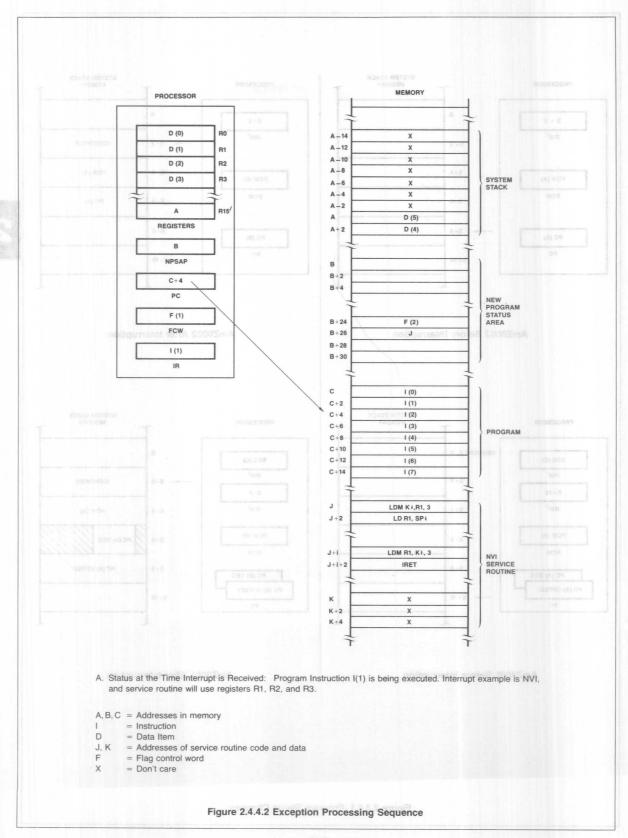


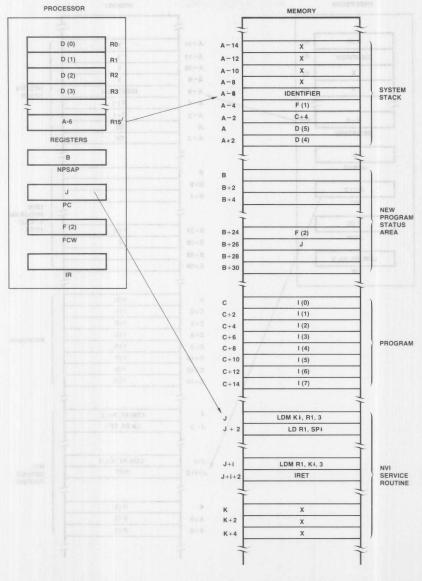
Figure 2.4.3.2 AmZ8002 New Program Status Area

TABLE 2.4.3 LOCATION OF FCW AND PC FOR INTERRUPTS AND TRAPS

		AmZ8001	AmZ8002		
Function	FCW	потнияльне РС	FCW	PC	
Special Opcode Trap	NPSAP + 10	NPSAP + 12, 14	NPSAP + 4	NPSAP + 6	
Privileged Instruction Trap	NPSAP + 18	NPSAP + 20, 22	NPSAP + 8	NPSAP + 10	
System Call Trap	NPSAP + 26	NPSAP + 28, 30	NPSAP + 12	NPSAP + 14	
Segment Error Trap	NPSAP + 34	NPSAP + 36, 38	Unused	Unused	
Nonmaskable Interrupt	NPSAP + 42	NPSAP + 44, 46	NPSAP + 20	NPSAP + 22	
Nonvectored Interrupt	NPSAP + 50	NPSAP + 52, 54	NPSAP + 24	NPSAP + 26	
Vectored Interrupt (V = Vector #)	NPSAP + 58	NPSAP + 4V + 60, 62	NPSAP + 28	NPSAP + 2V + 30	







B. Status after Interrupt is Acknowledged: Next instruction fetch will be at J. That instruction will save R1, R2, and R3 at K. The next instruction will copy the identifier into R1.

A, B, C = Addresses in memory

= Instruction

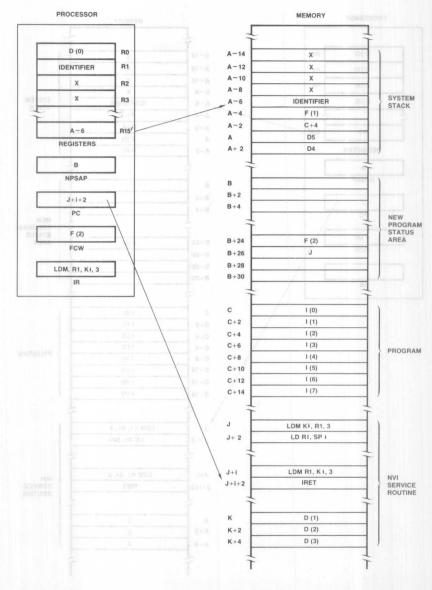
D = Data Item

J, K = Addresses of service routine code and data

F = Flag control word

X = Don't care

Figure 2.4.4.2 Exception Processing Sequence (Cont.)



C. Status Prior to End of Interrupt: Current Instruction will store registers; next instruction will return.

A, B, C = Address in memory

I = Instruction

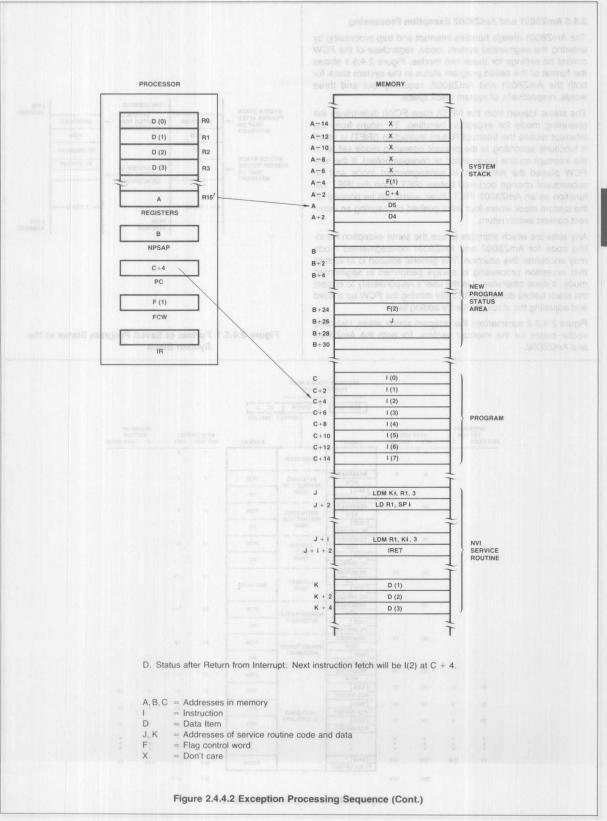
D = Data Item

J, K = Addresses of service routine code and data

F = Flag control word

X = Don't care

Figure 2.4.4.2 Exception Processing Sequence (Cont.)



2.4.5 AmZ8001 and AmZ8002 Exception Processing

The AmZ8001 always handles interrupt and trap processing by entering the segmented system mode, regardless of the FCW control bit settings for these two modes. Figure 2.4.5.1 shows the format of the saved program status in the system stack for both the AmZ8001 and AmZ8002, requiring four and three words, respectively, of system stack space.

The status loaded from the NPSA (new FCW) determines the operating mode for exception handling. To return from the interrupt routine the Interrupt Return instruction (IRET) is used. It functions according to the present operating mode set up in the interrupt routine (segmented or nonsegmented). If the new FCW placed the AmZ8001 in nonsegmented mode and no subsequent change occurred before IRET, then the IRET will function as an AmZ8002 IRET; three words will be popped off the system stack where four were pushed on, causing an incorrect context switch return.

Any software which attempts to use the same exception handling code for AmZ8002 and AmZ8001 nonsegmented mode may encounter this situation. The general solution is to ensure that exception processing is always performed in segmented mode. If done otherwise, it is the user's responsibility to correct the stack before doing an IRET (by moving the FCW up a word and adjusting the stack pointer by adding two).

Figure 2.4.5.2 summarizes the program status areas, including vector codes for the interrupt vectors, for both the AmZ8001 and AmZ8002.

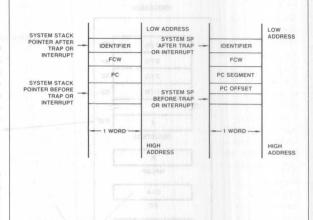


Figure 2.4.5.1 Format of Saved Program Status in the System Stack

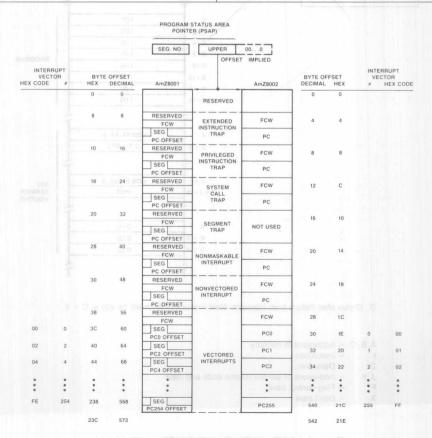


Figure 2.4.5.2 Program Status Area

2.5 STATUS LINES

The AmZ8000 CPU outputs status information over its four status lines (ST0-ST3) and the System/Normal line (S/N). This information can be used to extend the addressing range or to protect accesses to certain portions of memory. The types of status information and their codes are listed in Table 2.5.

Status conditions are mutually exclusive and can, therefore, be encoded without penalty. Most status definitions are self-explanatory. One code is reserved for future enhancements of the AmZ8000 Family.

Extension of the addressing range is accomplished in an AmZ8000 system by allocating physical memory to specific usage (code vs. data space, for example) and using external circuitry to monitor the status lines and select the appropriate memory space for each address. For example, the direct addressing range of the AmZ8002 CPU is limited to 64K bytes; however, a system can be configured with 128K bytes if additional logic is used, say, to select the lower 64K bytes for program references and the upper 64K bytes for data references.

Protection of memory by access types is accomplished similarly. The memory is divided into blocks of locations and associated with each block is a set of legal status signals. For each access to the memory, the external circuit checks whether the CPU status is appropriate for the memory reference. The AmZ8010 Memory Management Unit is an example of an external memory-protection circuit, and it is discussed in section 2.2.4.

The first word in an instruction fetch has its own dedicated status code, namely 1101. This allows the synchronization of external circuits to the CPU. During all subsequent fetch cycles within the same instruction (the longest instruction requires a total of four word fetches), the status is changed from 1101 to 1100. Load Relative and Store Relative also have a status of 1100 with the data reference, so information can be moved from program space to data space.

TABLE 2.5 STATUS DECODES

ST3-ST0	Definition			
0000	Internal operation			
0001	Memory refresh			
0010	I/O reference			
0011	Special I/O reference			
0100	Segment trap acknowledge			
0101	Nonmaskable interrupt acknowledge			
0110	Nonvectored interrupt acknowledge			
0111	Vectored interrupt acknowledge			
1000	Data memory request			
1001	Stack memory request			
1010	Data memory request (EPU)			
1011	Stack memory request (EPU)			
1100	Instruction space access			
1101	Instruction fetch, first word			
1110	Extension processor transfer			
1111	Reserved			

2.6 MEMORY AND I/O ADDRESSING

Like most 16-bit microprocessors, the AmZ8000 CPU uses a 16-bit parallel data bus between the CPU and memory or I/O. The CPU is capable of reading or writing a 16-bit word with every access. Words are always addressed with even addresses (A0 = 0). All instructions are words or multiple words and are aligned on even byte boundaries of memory.

The AmZ8000 CPU can, however, also read and write 8-bit bytes, so memory and I/O addresses are always expressed in bytes. The Byte/Word (B/W) output indicates whether a byte or word is addressed (High = byte). A0 distinguishes between the upper and lower byte in memory or I/O. The most significant byte of the word is addressed when A0 is Low. Refer to Figure 2.6.

For word operations in both the read and write modes, $B/\overline{W} = Low$, A0 is simply ignored and A1-A15 address the memory or I/O. For byte operations in the read mode, $B/\overline{W} = High$, A0 is again ignored, and a whole word (both bytes) is read, but the CPU internally selects the appropriate byte. For byte operations in the write mode, the CPU outputs identical information on both the low (AD0-AD7) and the high (AD8-AD15) bytes of the Address/Data bus. External hardware must be used to enable writing in one memory byte and at the same time disable writing in the other byte, as defined by A0. The replication of byte information for writes is for current implementation and may change for subsequent AmZ8000 CPUs; therefore system designs should not depend upon this feature.

I/O transfers between CPU and I/O devices are peformed with 8-or 16-bit transfers. I/O devices are addressed with a 16-bit I/O port address. (Segment addresses are not involved.) The I/O port address is similar to a memory address; however, I/O address space is not part of the memory address space. I/O port and memory addresses co-exist on the same bus and they are distinguished by the status outputs.

Two types of I/O instructions are available: standard and special. Each has its own address space. Standard I/O instructions include a comprehensive set of In, Out and Block I/O instructions for both bytes and words. Special I/O instructions are used for loading and unloading the Memory Management Unit. The status outputs distinguish between standard and special I/O references.

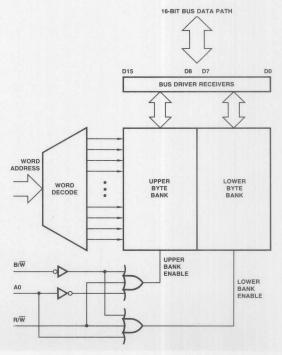
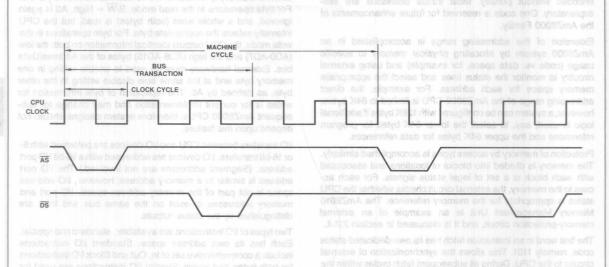


Figure 2.6 Byte/Word Selection of Memory

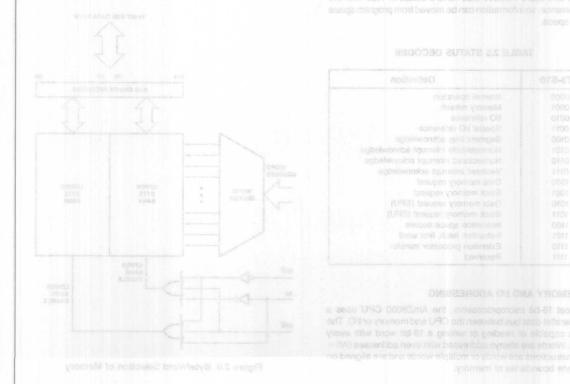
2.7 TIMING The baser oals, however, has USO 00085mA edit

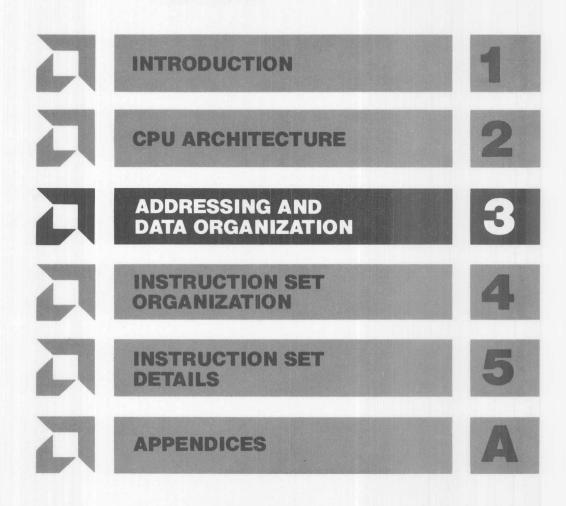
Figure 2.7 shows the three basic timing periods of the AmZ8000: a clock cycle, a bus transaction, and a machine cycle. A clock cycle (sometimes called a T-state) is one cycle of the CPU clock, starting with a rising edge. A bus transaction covers a

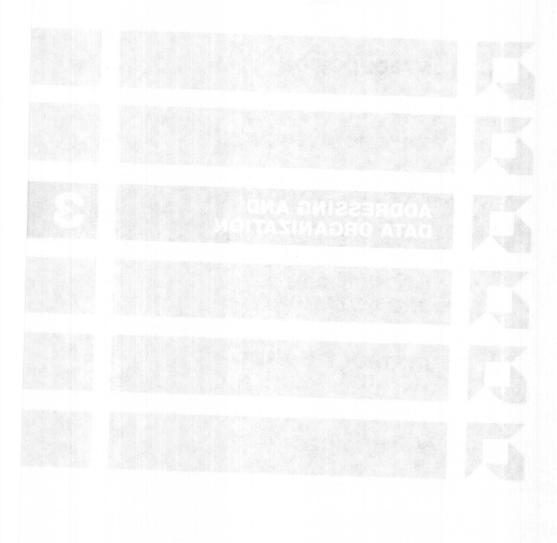
single data movement on the CPU bus and will last for three or more clock cycles, starting with a falling edge of \overline{AS} and ending with a rising edge of \overline{DS} . A machine cycle covers one basic CPU operation and always starts with a bus transaction. A machine cycle can extend beyond the end of a transaction by an unlimited number of clock cycles.



auta te ad T. fint inamaga half gomes d'art pobas. Figure 2.7 Basic Timing Periods attuation not contrait teagnet art) noticentent amagant de l'apprendict de la contrait d







3.0 ADDRESSING AND DATA ORGANIZATION

3.1 INTRODUCTION

Operands in the AmZ8000 may be part of the instruction, in registers, or in memory. In any case they may be bytes, words or long words. For operands in memory they may also be addressed as strings — sequences of bytes or words up to 64K bytes long. The type of operand, byte, word, long-word or string, is determined by the instruction. Most instructions have both byte and word forms. (The mnemonic for the word form of the instruction is the basic name. The byte form appends a B to the name and the long-word form appends an L.) For each instruction there is one or more possible address modes used to designate the location of the operand. The general form of the various modes and detailed descriptions follow later. Addressing modes are also discussed in detail.

3.2 ADDRESSING DATA

Data can be addressed and stored in the general-purpose registers, in memory, or in instructions. Because the registers are general-purpose, addresses can also be easily manipulated as data. Bottlenecks due to information exchanges between dedicated data and address registers do not exist.

3.2.1 Addressing Data in Registers

Instructions refer to data in registers using the **R** (Register) address mode. The opcode specifies byte, word or long word, and the appropriate register or register pair is referenced as shown in Table 2.3.1 and Table 3.2.1.

3.2.2 Addressing Data in Memory

Data located in memory is referenced by supplying in the instruction one of the following:

- a. The complete address DA (direct address) mode.
- The name of a register or register pair containing the complete address – IR (indirect register) mode.
- A complete address and the name of a register whose contents is to be added to the specified address X (indexed) mode
- d. A displacement and the name of a register or register pair containing a complete address to which the displacement should be added to that address – BA (base indexed) mode.
- The name of a register or pair containing a complete address and the name of a register containing a displacement to be added to that address – BX (base indexed) mode.
- f. A displacement which is to be added to the Program Counter
 RA (relative address) mode.
- g. Two registers, one containing the address of the beginning or end of a series of data items and the other containing the length of the series. Following each iteration of this type of instruction the address is changed to point to the next item and the count is decremented. This type of addressing is used for string manipulation and block I/O. Some instructions use a third register which points to a second string of data items to be processed with the first.

TABLE 3.2.1 DATA ADDRESSING IN REGISTERS

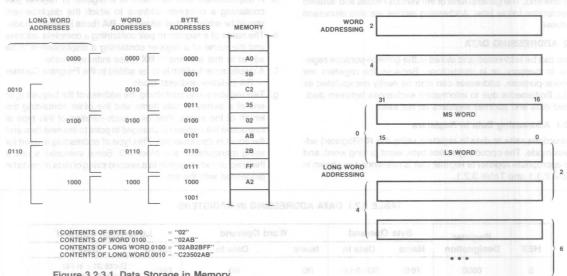
	Register	Byte	Byte Operand		Word Operand	Lor	ng-Word Operand	
HEX	Designation	Name	Data In	Name	Data In	Name	Data In	
0	0000	RH0	R0<8:15>	R0	R0	RR0	D<16:31> in R0 D<0:15> in R1	
1	0001	RH1	R1<8:15>	R1	R1		Reserved	
2	0010	RH2	R2<8:15>	R2	geusge (R2 amo) aidt	RR2	D<16:31> in R2 D<0:15> in R3	
3	0011	RH3	R3<8:15>	R3	R3 s ristnisr	ortant to p	Reserved	
4	0100	RH4	R4<8:15>	R4	lo end 184 pin ent ma	RR4	D<16:31> in R4 D<0:15> in R5	
5	0101	RH5	R5<8:15>	R5	R5	Mish to so	Reserved	
6	0110	RH6	R6<8:15>	R6	R6	RR6	D<16:31> in R6 D<0:15> in R7	
7	0111	RH7	R7<8:15>	R7	ritod an R7 pol viorne	secutive s	Reserved	
8	1000	RL0	R0<0:7>	R8	R8	RR8	D<16:31> in R8 D<0:15> in R9	
9	1001	RL1	R1<0:7>	R9	R9 10 8810	ides the ad	Reserved	
А	1010	RL2	R2<0:7>	R10	R10	RR10	D<16:31> in R10 D<0:15> in R11	
В	1011	RL3	R3<0:7>	R11	R11	or long set	Reserved	
С	1100	RL4	R4<0:7>	R12	R12	RR12	D<16:31> in R12 D<0:15> in R13	
D	1101	RL5	R5<0:7>	R13	fid to 91 R13 0 811 A	ary notation	Reserved	
E	1110	RL6	R6<0:7>	R14	R14	RR14	D<16:31> in R14 D<0:15> in R15	
F pro	penth 1111 abni e	RL7	R7<0:7>	R15	R15 in normal mode R15' in system mode (SP for AmZ8002)	RR14'	Data in R14' and R15 in system mode (SP for AmZ8001)	

(Register quads are not shown in the table. They are designated as RQ0, RQ4, RQ8, and RQ12.)

3.2.3 Data Storage in Memory

Memory address space is viewed to a chain of consecutively numbered (in ascending order) bytes, as shown in Figure 3.2.3.1. The number of each byte is its address, and the byte is the basic addressable element. A word spans two bytes, and is addressed by the address of its high order byte (most significant), with the lowest absolute address of the two bytes, which is always an even address. A long-word consists of four bytes and is also referred by the address of its high order byte (most significant word), which is the lowest absolute address of the four bytes.

Instructions and addresses stored in memory are always on word boundaries; they always have even numbered addresses.



BYTE

MS

BYTE

18

BYTE

ADDRESSING

Figure 3.2.3.1 Data Storage in Memory

Note that this format differs from the PDP-11 but is identical to the IBM convention. The reason for choosing this format is because the AmZ8000 CPU can operate on 32-bit long words and also on byte and word strings. It is important to maintain a continuity of order when words are concatenated into long words and strings. Making ascending addresses proceed from the highest byte of the first word to the lowest byte of the last word maintains this continuity, and allows comparing and sorting of byte and word strings. Refer to Figure 3.2.3.2.

String instructions, such as I/O and block compare, refer to a series of bytes or words in consecutive memory locations. Both autoincrement and autodecrement forms of these instructions exist, so the string can be scanned starting at either end. The byte form of these instructions modifies the address by one on each iteration to point to the next byte; the word form modifies the address by two to point to the next word.

Bit labeling within a byte does not follow this order. The least significant bit in a byte, word or long word is called Bit 0 and occurs in the byte with the highest memory address. This is consistent with the convention where bit n corresponds to position 2n in the conventional binary notation. This ordering of bit numbers is also followed in the registers.

3.2.4 Data Contained in Instruction

Most instructions allow an operand to be contained within the instruction itself. This is the IM (immediate) address mode. Generally, the data follows the opcode and any addresses which are also part of the instruction.

Figure 3.2.3.2 Memory Addressing

Immediate data may consist of a 16-bit word, long-word, or a byte. The long-word has the more significant 16 bits first, then the less significant 16 bits in the second word. A byte operand uses a full 16-bit word with the same data in both the upper and lower bytes.

There are a few instructions which permit immediate data to be located directly in the single 16-bit word which contains the opcode, forming a one-word instruction. These include a Load Byte instruction, and Increment or Decrement by any integer from 1 through 16.

3.3 MEMORY ADDRESS FORMATS

A complete memory address may take one of three forms. Refer to Figure 3.3.

3.3.1 NS (Nonsegmented)

A single 16-bit word which specifies the address in the AmZ8002 or in the nonsegmented mode of the AmZ8001.

3.3.2 SSO (Segmented Short Offset)

A single 16-bit word in the form shown having a shortened offset address, but including a segment number.

This form can address a byte in the first 256 bytes of any segment. The range is effectively extended to the entire memory space when this form is used with the indexed addressing mode. This form offers the ability to access data in any segment of memory without using two full words to supply the address.

The short offset segmented address can be used only in direct address and indexed addressing modes where the address is part of the instruction.

3.3.3 SLO (Segmented Long Offset)

Two 16-bit words in the form shown completely specifying both the 16-bit offset and the 7-bit segment addresses.

This form can refer directly to any byte in the memory. The word containing the segment always precedes the word containing the offset. When a register pair is used to hold this form, the segment is in the even numbered register and the offset is in the the odd numbered register of the pair. (For example, in RR2, R2 contains the segment and R3 contains the offset.)

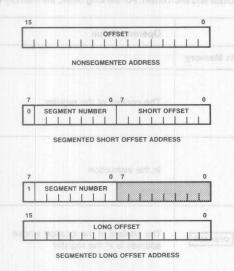


Figure 3.3.1 Memory Addressing Formats

In the general unrestricted case of long offset, the segmented address occupies two words, as described before. The most significant bit in the segment word is a one in this case.

The short offset mode squeezes the segment number and offset into one word, saving program size and execution time. Since 23 bits obviously do not fit into a 16-bit word, the eight most significant bits of the offset are omitted and implied to be zero. The most significant bit of the address word is made zero to indicate short offset mode. Short offset addresses are thus limited to the first 256 bytes at the beginning of each segment. At first this may appear to be a restriction, but it is very useful when used with the index mode, where the index register can always supply the full 16-bit range of the offset. Short offset saves one instruction word and speeds up execution by two clock cycles in direct address mode and three clock cycles in indexed mode.

3.4 SEGMENTED ADDRESS FORMATS

Figure 3.4 shows the format for segmented addresses.

3.5 DATA TYPES

Operands are 1, 4, 8, 16, 32 or 64 bits, as specified by the instruction. In addition, strings of 8- or 16-bit data can be manipulated by single instructions. Of particular interest are the increased precisions of the arithmetic instructions. Add and Subtract instructions can operate on 8-, 16-, or 32-bit operands; Multiply instructions can operate on 16-bit or 32-bit multiplicands; and Divide instructions can operate on 32- or 64-bit dividends. The Shift instructions can operate on 8-, 16-, and 32-bit registers. Figure 3.6.1 Decoding Address Modes

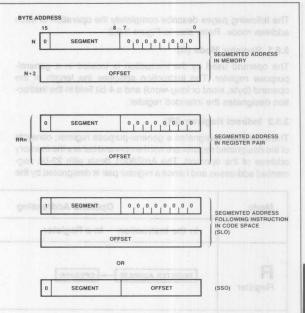
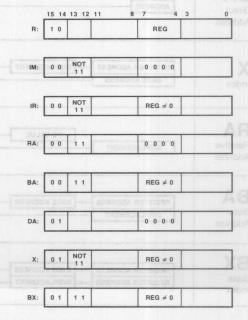


Figure 3.4 Segmented Address Formats

3.6 ADDRESS MODES

In addition to supporting the primitive operands of bits, digits, bytes, 16- and 32-bit integers, and byte and word strings, the AmZ8000 CPU's rich set of addressing modes supports highlevel data constructs such as arrays, lists, and records. These combine with powerful instructions to significantly extend the capabilities of microprocessors.

The address mode for a given instruction is determined by certain bits in the instruction, as shown in Figure 3.6.1. Refer to the format and decoding sections of the instruction set organization chapter.



The following pages describe completely the operation of each address mode. Refer also to Figure 3.6.2.

3.6.1 Register Mode (R)

The operand used by the instruction is located in a generalpurpose register. The instruction specifies the length of the operand (byte, word or long-word) and a 4-bit field in the instruction designates the intended register.

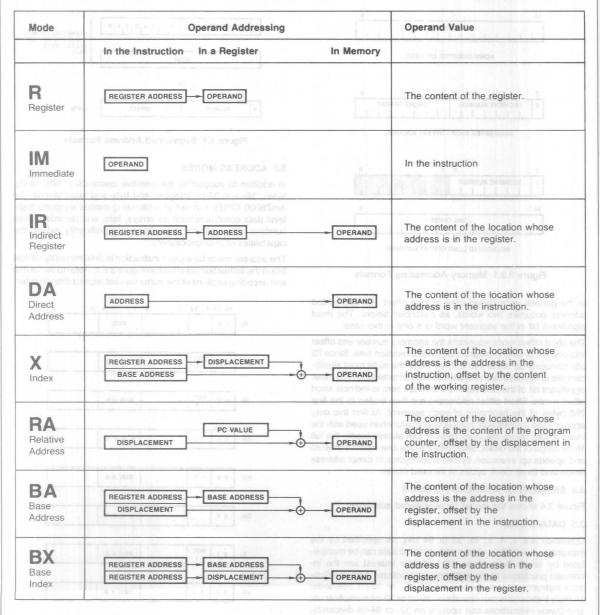
3.6.2 Indirect Register (IR)

The instruction designates a general-purpose register; contents of the designated register are not the operand but are the memory address of the operand. The AmZ8001 deals with 23-bit segmented addresses and hence a register pair is designated by the

instruction (in segmented mode). The first register contains the 7-bit segment number and the second register contains the 16-bit offset. Any general-purpose register pair except RR0 can be designated for this addressing mode. The AmZ8002 requires only 16-bit addresses and hence any general-purpose register except R0 can be designated for IR addressing mode. (Some exceptions to this are noted in the instruction set description. See also Autoincrement and Autodecrement.)

3.6.3 Direct Address (DA)

The instruction itself explicitly specifies an address and the operand used by the instruction is located at that address. In the AmZ8001, direct addresses are specified in one of two formats – long offset and short offset. For the long offset, the memory word



addisM and the Application of the Figure 3.6.2 Addressing Modes

immediately following the instruction opcode word contains the 7-bit segment number, and the memory word immediately following the segment number word is the 16-bit offset. For the short offset, the memory word immediately following the instruction opcode word contains both 7-bit segment number and 8-bit offset. In the AmZ8002 the memory word immediately following the instruction opcode word contains the 16-bit address.

3.6.4 Immediate Mode (IM)

The instruction itself contains the operand. In certain short instructions the operand and opcode are in one word. In general, the operand is in the last word or words of the instruction. Byte operands are repeated in both halves of the word.

3.6.5 Indexed Mode (X)

The instruction designates a 16-bit general-purpose register as the index register. Any general-purpose register except R0 can be used as the index register. The instruction also specifies an address as in the direct address mode. In the AmZ8001 the 16-bit contents of the designated index register are added to the 16-bit offset value specified in the instruction. Both index and offset are treated as 16-bit unsigned integers and any carry from the most significant bit position during this addition is ignored. The resulting 16-bit sum together with the 7-bit segment number specified in the instruction is used as 23-bit segmented address. The operand will be located at this address in memory. If short addressing offset is used in the AmZ8001 for indexed addressing mode, the memory word immediately following the instruction opcode word contains both a 7-bit segment number and an 8-bit offset.

A 16-bit unsigned integer is formed whose least significant byte is the 8-bit offset specified and most significant byte is zero. The 16-bit word thus formed is added to the 16-bit unsigned integer contained in the designated general-purpose register. Any carry from the most significant bit position during this addition is ignored. The 16 bits resulting from this addition together with the 7-bit segment number specified is the 23-bit address. The operand will be located in the memory at this address.

In the AmZ8002 the memory word immediately following the instruction opcode word contains a 16-bit address. This unsigned integer is added to the 16-bit unsigned integer located in the designated index register. The carry from the most significant bit position during this addition is ignored. The resulting 16-bit address is where the operand is located in the memory.

3.6.6 Base Address Mode (BA)

The instruction designates a general-purpose register as the base address register. In the case of the AmZ8001 the instruction designates a register pair such that the 7-bit segment number is contained in one register and the 16-bit offset is contained in the other as shown. In the case of the AmZ8002 the designated base address register contains a 16-bit address. Any general-purpose register except R0 or register pair except RR0 can be designated as the base address register. The memory word immediately following the instruction opcode word contains a 16-bit displacement. Both displacement and base address are treated as unsigned binary integers. The 16-bit displacement is added to the 16-bit base address (16-bit offset in the AmZ8001) and carry occuring from the most significant bit position during this addition is ignored. The resulting 16-bit value (together with the segment number of the base address in the AmZ8001) is the address of the operand in memory.

3.6.7 Base Indexed Mode (BX)

The instruction designates a general-purpose register (register pair in AmZ8001) as the base address register. The instruction also designates a 16-bit general-purpose register as displace-

ment. Any general-purpose register except R0 (AmZ8002) or any register pair except RR0 (AmZ8001) can be used as the base address register. Similarly any general-purpose register except R0 can be used as the displacement register. Both base address and displacement are unsigned integers.

The 16-bit displacement is added to the base address (or offset of the base address in AmZ8001) and carry from the most significant bit position during this addition is ignored. The 16-bit result (together with base address segment number) is the address of the operand in memory.

3.6.8 Relative Address (RA)

The instruction itself contains a displacement. This displacement is a signed integer using two's complement notation. The number of bits allocated to represent the displacement depend on the instruction where relative addressing mode is available. The displacement is sign extended appropriately to obtain a signed 16-bit displacement. The sign extended displacement is added to the 16-bit program counter (PC offset in AmZ8001). Carry from the most significant bit position during this addition is ignored. As soon as the instruction using the relative address mode is fetched, the PC will be updated. Hence, the updated PC value (i.e., address of the following instruction) will be used for address calculations.

The 16-bit value obtained by adding the PC and displacement (together with the segment number in AmZ8001) is the address of the operand in memory.

3.6.9 Autoincrement and Autodecrement

These two implied addressing modes are only used in string manipulation instructions. These addressing modes are a variation of the IR addressing mode. The instruction designates a general-purpose register (or a register pair in AmZ8001) whose contents are used as the address. After fetching the operand the contents of the register are incremented or decremented depending on Autoincrement or Autodecrement. In the case of the AmZ8001, only the register containing the offset is affected and any carry resulting from this operation is ignored. For byte operations, incrementing or decrementing by one occurs. For word operations, incrementing or decrementing by two takes place. R0 and RR0 can be designated in the IR addressing mode for the autoindexing instruction.

3.6.10 Port Addressing Modes

Input/output transfers between CPU and I/O devices are performed with 8- or 16-bit transfers. The address of the I/O ports is similar to a memory address in that they coexist on the same bus; however, they are in separate address spaces distinguished by the status outputs.

I/O devices are addresses with a 16-bit I/O port address. Segment addresses are not involved. Two types of I/O instructions are supported with separate I/O spaces distinguished by the status outputs. These instructions use two types of port addressing modes:

Port register (PR): The instruction designates a general-purpose register (never a register pair) which contains the port address (similar to IR memory addressing mode).

Port Address (PA): The instruction contains an explicit port address word which specifies the I/O port address (similar to DA memory addressing mode).

The autoindexing versions of the I/O instructions combine port register port addressing with the indirect register memory addressing mode.

Immediately following the instruction opcode word contains the 7-bit segment number, and the training word immediately following the segment number word is that 1-bit office. For the short offset, the memory word immediately allowing the instruction opcode word contains both 7-bit segment number and 8-bit offset in the And 2-5022 the memory, word into exhabit following the instruction opcode word contains the 16-bit address.

3.6.4 immodiate Mode (IM)

The instruction fixed politake the ope and, to defait short instructions are operand and oppode are in one word to general, the operand is in the fast word or word so the instruction. Byte operands are represed in both helives of the front.

20 oboth besset 2.a.c.

The instruction designature at 16-bit general ourgose register assept PO can the index register. Any general-purpor a register assept PO can be used as the index register. The instruction also apocities an address a finishe defect address mode in the en/2000 the re-bit contents of the designated index register are added to the re-bit offset value apocitied in the instruction Both index and offset are specified in the instruction Both index and offset are specified in the instruction Both index and offset are specified in the position of the first of the present number specified in the transition is used as 25-bit segment address. The operand will be remained at this address in memory if short addressing make the memory world interesting the memory sold in the Am-ZSDOT for indicate and an 8-bit offset.

A 18 bit shaighed integer is formed which have as past significant byte as the 8-bit offset specified and most organisant byte is zero. The 16-bit word thus formed is added to the 16-bit unsigned integer contrared in the cost presed bendral suppose rejector. Any carry from the most significant bit position during this addition as ignored. The 16-bits resulting from the addition apastrar with the contrared number specified is the 23-bit address. The contrared will be forceded in the present by at this address.

In the Amizgotz the memory were immediately following the instruction operate word contains and bit address. This unsigned interest is added to the 16-bit unsigned integer located in the deric nated index register. The party from the most significant bit position during this addition is unto at the estitling 16-bit address is where the operated is located in the integer.

(All) about assetted and a an

The instruction designates a general-purpose register as the bass addition matter. In the case of the Am-ZBOOT fire instruction outsides a socialist pair such that the 7-bit segment number is contained in one orgist man the 75-bit offset is contained in the object and the 75-bit offset is contained in the others mattern and the Am-ZBOOT fire designated assess address register contains a 4-bit authess Ammanders purpose the base address register contains a 16-bit displaced as the contains a 16-bit displaced man flow offset and boar displacement and boar address are realted to the second binary integers. The 15-bit displacement is address to real boar displacement and boar address are realted to the outside binary integers. The 15-bit displacement is address to the outside bit base address. The result is position during the man flow has a price of the outside of the outside of the contains of the result of the contains of the result of the contains the result of the contains of of the

767 See administration of the Color

The instruction dealgrams a general-purpose rentate (register pair in Am 25001) as the bias ad tress register. The instruction as a second confidence of the second confidence as displace.

ment. Any general purchase rays her except (R04AmZ8002) or any register pair except R5-3 (Am2.2.0.01) can be used as the base address register Semillerly and prerel-purpose register seminary. P0 can be used as the displacement in gister, Both dase address and displacement are unsignal in tegers.

The 16-bit displacement is added to the page suidcass (in critical to be been address in Ar (7500)) and carry light the total significant bit position during the pastion is ignored. The hebit it as (regardered efficients) is the address of the opening the memory.

LARI Southtwo Address (FA)

The instruction itself contains a displacement. This displacement as signed integer using two's complement notation. The number of bits allocated to represent the displacement depend on the estimation where notative unit meting modells available. The displacement is sign extended appropriately to obtain a solution. The displacement. The sign extended appropriately to obtain a solution of the representation of the content of the sign of the relative and a factorial produced the produced as a feature of the produced of the sign of the sinterest sign of the sign of the sign of the sign of the sign of t

First 15-bit value obtained for evaling the PC and displacement to getting with the segment, or obserts Am28000 is the didness of the coefficient.

hisnestable to hametomonia 0.9.0

These two implied addressing orders are only used in study manapulation instructions. From a addressing modes are to value tool of the IR activessing mode. The instrugion designates a non of the IR activessing mode. The instrugion designates a contents are to as the patients after fatching the Arg20001) whose contents of the register are incremented or instrumented designating on Autoria, are incremented or instrumented designating on Autoria, are register containing the offset and single and the register containing the offset and single one. Incrementing or declaration are supported by two taxes present and RRO can be a sequenced in the IR addressing mode for the enterprise of the life and RRO can be a sequenced in the IR addressing mode for the

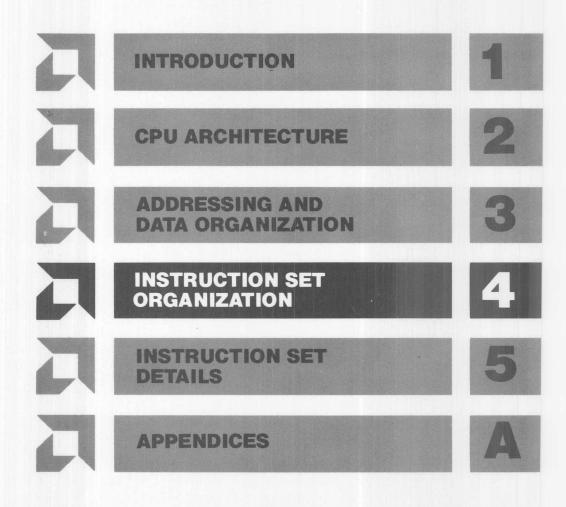
salts Port Addressing Glodes

injurroupus maiseiers permenn umb and incipilities and not go possible formad with 8-or 15-bit transfers. The address pilities UC possible similar to a memory actions to their they constituted the finite bussing ware in separate address spaces distinguished by the status outputs.

I/O devices are authorized with a Verill I/O port address. Sand ment addresses are unit violed. Two types of I/O inconsitions are supposed with separate I/O appage distinguished by the status outputs. These took units use two-types of port addressing medical.

Pontegister (PR): The us midlen designates argument purpose regarders there is equal to purpose sentials designate to the memory addressing meda).

For address (RA). The establish contains an applica port address word which specifies the VO port accress (similar to DA menors addressent mode).



4.0 INSTRUCTION SET ORGANIZATION

4.1 INTRODUCTION

The AmZ8000 offers an abundant instruction set that represents a major advance over its predecessors. The Load and Exchange instructions have been expanded to support operating system functions and conversion of existing microprocessor programs. The usual Arithmetic instructions can now deal with higher-precision operands, and hardware Multiply and Divide instructions have been added. The Bit Manipulation instructions can access a calculated bit position within a byte or word, as well as specify the position statically in the instruction.

The Rotate and Shift instructions are considerably more flexible than those in previous microprocessors. The String instructions are useful in translating between different character codes. Special I/O instructions are included to manage peripheral devices, such as the Memory Management Unit (AmZ8010), that do not respond to regular I/O commands. Multiple-processor configurations are supported by special instructions.

The following instructions exemplify the innovative nature of the AmZ8000 instruction set. A complete list of AmZ8000 instructions can be found in the Instruction Set section.

Load and Exchange Instructions

Exchange Byte (EX) is practical for converting Z-80, 8080, 6800 and other microprocessor programs into AmZ8000 code, because the AmZ8000 uses the opposite assignment of odd/even addresses in 16-bit words.

Load Multiple (LDM) saves n registers and is useful for switching tasks.

Load Relative (LDR) loads fixed values from program space into data space.

Arithmetic Instructions

Add With Carry and Subtract With Carry (ADC, SBC) are conventionally used in 8-bit microprocessors for multiprecision arithmetic operations. These instructions are rarely used with the AmZ8000 CPU because it has 16- and 32-bit arithmetic instructions.

Decrement By N and Increment By N (DEC, INC) are intended for address and pointer manipulation, but can also be used for Quick Add/Subtract Immediate with 4-bit nibbles. The flag setting is different from Add/Subtract instructions — as is conventional — in that the Carry and Decrement Adjust flags are unaffected by the Increment and Decrement instructions to support multiple precision arithmetic.

Decimal Adjust (DAB) automatically generates the proper 2-digit BCD result after a byte Add or Subtract operation, and eliminates the need for special decimal arithmetic instructions.

Multiply (MULT) provides signed (two's complement) multiplication of two words, generating a long-word result; or of two longwords generating a quadruple word result. No byte multiply exists because it is rarely used and, after sign extension, can be performed by a word multiply.

Divide (DIV) provides signed (two's complement) division of a long word by another word, generating a word quotient and a remainder word; or of one quadruple-word by a long-word, generating a long-word quotient and long-word remainder.

Both Multiply and Divide use a conforming register assignment. That is, a multiply followed by a divide on the same registers is essentially a no-op. The register designation used in the operating description must be even for word operations and must be a multiple of four for long-word operations.

Logical Instructions and Instructions deliberated the contractions and the contractions and the contractions are contracted to the contractions are contracted to the contractions are contracted to the contraction of the contra

Test Condition Code (TCC) performs the same test as a Jump instruction, but affects the least significant bit of a specified register instead of changing the PC.

Program Control Instructions

Call Relative (CALR) is a shorter, faster version of Call, but with a limited range.

Decrement And Jump If Non-Zero (DJNZ) is a one-word basic looping instruction.

Jump Relative (JR) is a shorter, faster version of Jump, but with limited range.

Bit Manipulation Instructions

Test Bit, Reset Bit, Set Bit (BIT, RES, SET) are available in two forms: static and dynamic. For the static form, any bit (the position is defined in the immediate word of the instruction) located in any byte or word in any register or in memory can be set, reset or tested (inverted and routed into the Z flag).

For the dynamic form, any bit (the position is defined by the content of a register that is, in turn, specified in the instruction) located in any byte or word in any register, but not in memory, can be set, reset or tested.

Test And Set (TSET) is a read/modify/write instruction normally used to create operating system locks. The most significant bit of a byte or word in a register or in memory is routed into the S flag bit and the whole byte or word is then set to all 1s. During this instruction, the processor does not relinquish the bus.

Test Multi-Micro Bit and Multi-Micro Request/Set/Reset (MBIT, MREQ, MSET, MRES) are used to synchronize the access by multiple microprocessors to a shared resource, such as common memory, bus, or I/O device.

Note that the instruction MREQ (Multi-Microprocessor Request) has nothing whatsoever in common with the MREQ (Memory Request) output from the AmZ8000 CPU.

Rotate and Shift Instructions

The AmZ8000 CPU has a complete set of shift instructions that shift any combination of bytes or words, right or left, arithmetically or logically, by any meaningful number of positions as specified either in the instruction (static) or in a register (dynamic).

The CPU also has a smaller repertoire of rotate instructions that rotates bytes or words, either right or left, through carry or not, and by one bit or by two bits.

The instructions Rotate Digit Left and Rotate Digit Right (RDLB, RRDB) rotate 4-bit BCD digits right or left, and are used in BCD arithmetic operations.

Block Transfer and String Manipulation Instructions

Translate And Decrement/Increment (TRDB, TRIB) is used for code conversion, such as ASCII to EBCDIC. These instructions translate a byte string in memory by substituting one string by its table-lookup equivalent. TRDB and TRIB execute one operation and decrement the contents of the length register; thus they are useful as part of loop performing several actions on each character.

Translate, decrement/Increment and Repeat (TRDBB, TRIRB) are the same as TRDB and TRIB, except they repeat automatically until the contents of the length register become zero. They are therefore useful in straightforward translation applications.

Translate And Test, Decrement/Increment (TRTDB, TRTIB) test a character according to the contents of the translation table.

Translate And Test, Decrement/Increment And Repeat (TRTDRB, TRTIRB) scans a string of characters. The first character is tested and, depending on the contents of the translation table, the process stops or skips to the next character. Stopped characters can be used for further processing.

I/O and Special I/O Instructions Tollandard Ionia

The AmZ8000 CPU has two complete sets of I/O instructions: Standard I/O and Special I/O. The only difference is the status information on the ST0-ST3 outputs.

Standard I/O instructions are used to communicate with AmZ8000 bus compatible peripherals. Special I/O instructions are typically used for communicating with the Memory Management Unit.

Both types of instructions transfer eight or 16 bits and use a type of 16-bit addressing analogous to the AmZ8002 memory-addressing scheme: For word operations, A0 is always zero; in byte-input operations, A0 is used internally by the CPU to select the appropriate byte; in byte-output operations, the byte is duplicated in the high and low bytes of the address/data bus, and external logic uses A0 to enable the appropriate output device.

4.2 INSTRUCTION FORMAT

The CPU instructions are one to five words long, depending on the type of instruction and addressing mode. Instructions are located in memory and must be word aligned. The first word of an instruction always contains the opcode. Depending on the addressing mode, one or more words will follow the opcode word of an instruction. Figure 4.2 illustrates the general instruction word format. Some instructions contain fields that differ from the generalized format shown. All such variations can be ascertained by referring to the individual instruction descriptions found in later sections of this document.

4.3 INSTRUCTION DECODING

The Mode Field (bit 14 and bit 15), together with bits 12 and 13 and bits 4, 5, 6 and 7, determines the applicable addressing mode. Bit 8 of the opcode word specifies word or byte operand whenever applicable. Bits 4, 5, 6 and 7 normally designate a general-

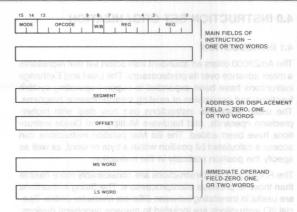


Figure 4.2 General Instruction Word Format

purpose register. Note that when designating a register pair, bit 4 must be zero and only 5, 6, and 7 are used. Refer to Figure 4.3.

For Register mode of addressing there are no restrictions on the values of bits 4, 5, 6 and 7. Only the mode field is needed to specify this addressing mode for any general-purpose register. However, for IM, RA and DA addressing modes, bits 4, 5, 6 and 7 must all be zero. For these addressing modes, zeroes in bits 4, 5, 6 and 7 are not interpreted as general-purpose register number zero. Similarly, for IR, BA, X and BX addressing modes, bits 4, 5, 6 and 7 cannot be zero. In other words, general-purpose register number zero cannot be used in these addressing modes. It should be emphasized that if a register pair is needed for these addressing modes, bit 4 is always zero and the non-zero requirement applies to bits 5, 6 and 7. (See also Section 5.2.4.)

4.4 SEGMENTED AND NONSEGMENTED MODES

In the AmZ8002 addresses are completely contained in a single 16-bit word. The AmZ8001 uses a segment and an offset, requiring two words to contain the 23-bit segmented address.

All the instructions have both segmented and nonsegmented forms. The only difference is the address references. The AmZ8001 has both segmented and nonsegmented modes of

Address Mode

Bits 15, 14	Bits 13, 12	Bits 7, 6, 5, 4	Mode
00	Not 11	0	IM
00	Not 11	Not 0	IR.
00	11	0	RA
00	fipid stateR on	Not 0	BA
01	ns bos Xel to t	br 800 0 mts no	DA
01	Not 11	Not 0	X
01	11	Not 0	BX
10	X	X	R
11	X	X	Special*

^{*}Used for short one-word instructions.

Word/Byte

Bit 8	Mode
0	Byte
and Sex	Word

Address, When Present

Displacement	16-Bit Word
Nonsegmented Address	16-Bit Word
Segmented Short Offset	0 + 7-Bit Segment + 8-Bit Offset
Segmented Long Offset	First Word: 1 + 7-Bit Segment + 8-Bit Unused Second Word: 16-Bit Offset

Immediate Data, When Present

Byte	Same Byte in Both Halves of Word
Word	16-Bit Word
Long-Word	First Word: Bits 16:31 of Operand Second Word: Bits 0:15 of Operand

Figure 4.3 Instruction Decoding

operation. The segment bit (15) of the FCW is used to enable the segmented mode. In the nonsegmented mode the AmZ8001 emulates the AmZ8002 so that code assembled for the nonsegmented AmZ8002 will execute on the segmented AmZ8001.

Since the hardware will be designed for the segmented processor, the AmZ8001 will continue to supply segment addresses even though the instructions do not contain them. The segment number address will be whatever it was prior to the switch to the nonsegmented mode. All memory accesses will be in the PC segment space.

Refer to section 2.2.3 for a general discussion of segmented memory addressing. Refer to Table 4.4 to see the distinctions of segmented and nonsegmented modes with respect to the addressing modes and the number of instruction words required.

4.5 CONDITION CODES

The Condition Code (CC) is a 4-bit field in some instructions that specifies certain flag settings. The operation performed by the instruction is in most cases determined by the outcome of comparing the actual flag settings with that specified by the CC field. Instructions that specify CC field include conditional jumps, return from subroutine and block/string manipulating instructions. The Condition Code definitions consist of true and false settings of the C, Z and P/V flags, signed and unsigned comparisons as shown in Table 4.5. One of the CC values specifies unconditional combinations in which flag settings are ignored.

4.6 INPUT/OUTPUT INSTRUCTIONS

A set of input/output (I/O) instructions is provided to perform 16-bit or 8-bit transfers including block transfers between the CPU and I/O devices. Input/Output devices are addresses using a 16-bit

Table 4.4 SEGMENTED AND NONSEGMENTED MODES

		Nonsegmented Mode		Segmented Mode
		AmZ8002	AmZ8001	AmZ8001
PC, SI	P and NPSAP	Standard	Segmented	Segmented
IR	Address Register	Rnotted	Rn anollous	to inherest CPU operations. For enRR ma
DA	Address Field	Word	Word	Word (SSO), Long Word (SLO)
X	Address Field Displacement (Index) Register	Word Rn	Word Rn	Word (SSO), Long Word (SLO) Rn
ВА	Displacement Field Base Address Register	Word Rn	Word Rn	Word RRn
вх	Base Address Register Displacement (Index) Register	Rn Rn	Rn Rn	RRn Rn Rn
RA	Dispacement*	Word	Word	Word flexients at called distribution of and

^{*}JR, CALR, DBJNZ, displacement field <16 bits.

TABLE 4.5 CC - FIELD DECODING

CC Field	Assembler Notation	Meaning	Flag Settings for CC True
1110	NZ NZ	Not Zero	Z = 0
0110	ZR	Zero	Z = 1
1111	NC	No Carry	C = 0
0111	CY	Carry BISD DAS BOUSE	C = 1
1100	PO	Parity Odd	P/V = 0
0100	PE	Parity Even	P/V = 1
1101	PL	Plus Ismatka ve bala	S = 0 1160 001 W 2110
0101	MI	Minus	S = 1 4 01122500191
1110	NE	Not Equal	Z = 0 Subject of the
0110	EQ ad ag	Equal o enclossient be	ded to include special
1100	NOV	Overflow is Reset	P/V = U og co bed solbe
0100	OV	Overflow is Set	P/V = 1
avert resp U	U; laat each EP	Signed Comparisons	entreon student sU93
1001	GE	Greater Than or Equal	S ⊕ P/V = 0
0001	LT	Less Than	S ⊕ P/V = 1
1010	GT	Greater Than	$Z + (S \oplus P/V) = 0$
0010	Leui l'Eusé eu	Less Than or Equal	Z + (S + P/V) = 1
		Unsigned Comparisons	
1111	LGE	Logical Greater Than or Equal	C = 0
0111	LLT	Logical Less Than	C = 1
1011	LGT	Logical Greater Than	$(C = 0) \cdot (Z = 0) = 1$
0011	LLE	Logical Less Than or Equal	C + Z = 1
1000		Unconditional (Always True)	_

Notes: • = AND + = OR ⊕ = EXCLUSIVE OR

address called "port address". Conceptually, the port address is very similar to a memory address. Logically, however, port address space is not a part of the memory address space. Although memory and port address information is physically transmitted on the same bus lines in hardware, means are provided to distinguish memory addresses from I/O addresses (using status output lines supplied by the CPU). Port address generation uses the same methodology that is used to generate operand addresses in the nonsegmented CPU using IR and DA addressing modes. In the Instruction Set section these are designated as Port Register (PR) and Port Address (PA) addressing modes.

Two types of I/O instructions are available — "standard I/O" and "special I/O". The address space used by the special I/O is logically separate from the standard I/O. Special I/O address space can be distinguished from the standard I/O space using the status output lines from the CPU. They are intended for communicating with the Memory Management Unit. The I/O instructions exist not only to transfer single words or bytes of data, but also blocks of data from contiguous memory locations.

4.7 INSTRUCTION PREFETCH (PIPELINING)

Most instructions conclude with two or three clock cycles being devoted to internal CPU operations. For such instructions, the subsequent instruction-fetch machine cycle is overlapped with the concluding operations, thereby improving performance by two or three clock cycles per instruction.

Examples of instructions for which the subsequent instruction is fetched while they complete are Arithmetic and Shift instructions.

Some instructions for which the overlap is logically impossible are the Jump instructions (because the following instruction location has not been determined until the instruction completes). Some instructions for which overlap is physically impossible are the Memory Load instructions (because the memory is busy with the current instruction and cannot service the fetch of the succeeding instruction).

4.8 EXTENDED INSTRUCTION PROCESSING

The AmZ8000 architecture has a mechanism for extending the basic instruction set through the use of external devices. Special opcodes have been set aside to implement this feature. When the CPU encounters instructions with these opcodes in its instruction stream, it will perform any indicated address calculation and data transfer, but otherwise treat the "extended instruction" as being executed by the external device. Fields have been set aside in these extended instructions which can be interpreted by external devices (called Extended Processing Units – EPUs) as opcodes. Thus, by using appropriate EPUs, the instruction set of the AmZ8000 can be extended to include specialized instructions.

In general, an EPU is dedicated to performing complex and time consuming tasks in order to unburden the CPU. Typical tasks suitable for specialized EPUs include floating-point arithmetic, data base search and maintenance operations, network interfaces, graphics support operations — a complete list would include most areas of computing. EPUs are generally designed to perform their tasks on data resident in their internal registers.

Moving information into and out of the EPU's internal registers, as well as instructing the EPU as to what operations are to be performed, is the responsibility of the CPU.

For the AmZ8000 CPU, control of the EPUs takes the following form. The AmZ8000 CPU fetches instructions, calculates the addresses of operands residing in memory, and controls the movement of data to and from memory. An EPU monitors this activity on the CPU's AD lines. If the instructions fetched by the CPU are extended instructions, all EPUs and the CPU latch the instruction (there may be several different EPUs controlled by one CPU). If the instruction is to be executed by a particular EPU, both the CPU and the indicated EPU will be involved in executing the instruction.

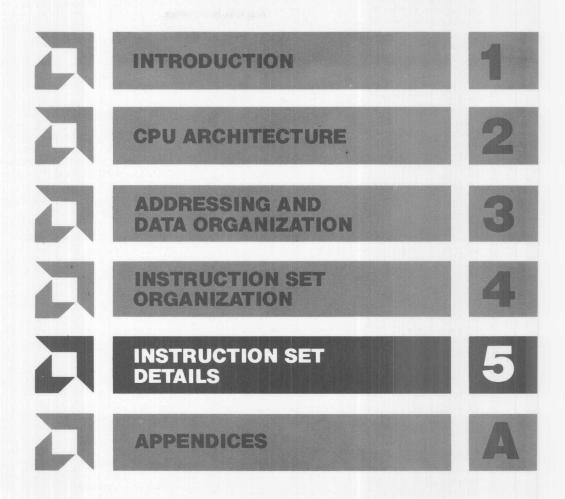
If the extended instruction indicates a transfer of data between the EPU's internal registers and the main memory, the CPU will calculate the memory address and generate the appropriate timing signals ($\overline{\text{AS}}$, $\overline{\text{DS}}$, MREQ, etc.), but the data transfer itself is between the memory and EPU (over the AD lines). If a transfer of data between the CPU and EPU is indicated, the sender places the data on the AD lines and the receiver reads the AD lines during the next clock period.

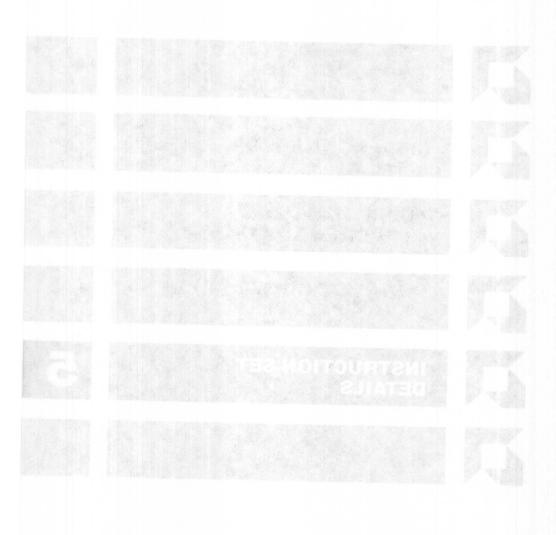
If the extended instruction indicates an internal operation to be performed by the EPU, the EPU begins execution of that task and the CPU is free to continue on to the next instruction. Processing then proceeds simultaneously on both the CPU and the EPU until a second extended instruction is encountered that is destined for the same EPU (if more than one EPU is in the system, all can be operating simultaneously and independently). If an extended instruction specifies an EPU still executing a previous extended instruction, the EPU can suspend instruction fetching by the AmZ8000 CPU until it is ready to accept the next extended instruction: the mechanism for this is the STOP line, which suspends CPU activity duing the instruction fetch cycle.

There are four types of extended instructions in the AmZ8000 CPU instruction repertoire; EPU internal operations; data transfers between memory and EPU; data transfers between EPU and CPU; and data transfer between EPU flag registers and CPU flag and control word. The last type is useful when the program must branch based on conditions determined by the EPU. Six opcodes are dedicated to extended instructions: 0E, 0F, 4E, 4F, 8E and 8F (in hexadecimal). The action taken by the CPU upon encountering these instructions is dependent upon an EPU control bit in the CPU's FCW. When this bit is set, it indicates that the system configuration includes EPUs; therefore, the instruction is executed. If this bit is clear, the CPU traps (extended instruction trap).so that a trap handler in software can emulate the desired operation.

In conclusion, the major features of this capability are, that multiple EPUs can be operating in parallel with the CPU, that the five main CPU addressing modes (Register, Immediate, Indirect Register, Direct Address, Indexed) are available in accessing data for the EPU; that each EPU can have more than 256 different instructions; and that data types manipulated by extended instructions can be up to 16 words long.

The extended processing instructions are included in Section 5.8 following the general instruction pages.





5.0 INSTRUCTION SET DETAILS

5.1 INTRODUCTION

This chapter provides detailed descriptions of the AmZ8001 and AmZ8002 instruction set. The instructions are listed by mnemonic in alphabetical order on the following pages.

The reader is referred to previous chapters for descriptions pertaining to items indicated on the detailed instruction pages, such as addressing modes, register designations, instruction format and decoding, flags and condition codes, system and normal instructions, and segmented and nonsegmented modes. The extended instructions discussed in previous sections are listed immediately following the detailed instruction pages.

Information regarding details of the instructions is given in the next sections. This includes a description of the assembler syntax shown used, a key to the instruction pages, and a summary of architectural data for quick reference in understanding these details

5.2 INSTRUCTION NOTATION AND ENCODING

The details for each general instruction on the following pages begin at the top of the page with the mnemonic and name of the instruction. A generic assembler statement for the instruction is shown. Also, system-only instructions are indicated, and in some cases, a reference to a similar instruction is indicated.

Below this information and to the right is given the operation which the instruction implements and a detailed verbal description of the instruction. Where necessary, a discussion of assembler notation is also given. At the bottom of the page the operation of the flags is given showing which flags are set, cleared, unaffected, or conditionally changed. The conditional changes are defined.

The left section of the page shows each specific form of the general instruction. This includes all available addressing modes and gives the format for both segmented and nonsegmented versions. Both segmented short offset and segmented long offset formats are given. Each specific form indicates the binary machine code with variable binary operand fields defined. Above the machine code representation is shown the specific assembler language syntax for that form. Also shown to the right of the machine code is the number of clocks required for execution of the instruction for that addressing mode.

5.2.1 Instruction Mnemonics

Each instruction page is listed by mnemonic in alphabetical order.

Instructions with byte, word, and long word data operands are described on separate pages. The mnemonic suffix "B" refers to a byte instruction, the suffix "L" refers to a long word instruction, and no suffix designates a word instruction. An example is the Shift Dynamic Logical instruction: SDLB, SDLL, and SDL, respectively. For some instructions, a data size either is not applicable or depends on the segmentation mode; here, the mnemonic does not have a suffix to indicate data size.

Some instructions have the relative address (RA) addressing mode. These are indicated with a mnemonic suffix "R." An example is LOAD Word into Register, LD and LDR. Note that the relative form is included on the general instruction page (LD) as well as being listed on a separate page (LDR).

The letter "R" is also used in the mnemonics to denote a repeat version of an autoindexing instruction. An example is the instruction INPUT Word from I/O Port to Memory, Autoincrement and Repeat (INIR).

5.2.2 Instruction Encoding

The binary encoding of the instruction is given for both segmented (including SSO and SLO where applicable) and non-segmented versions for each addressing mode. Fields specifying register operands, such as "Rbs," RRd," etc. and other operands, such as "n," are similar to the assembler language syntax description of the instruction. The binary encoding for the register fields is repeated as Table 5.5.1. Some restrictions on register fields are noted in the following sections.

In the case of nonsegmented instructions the designation "ADDRESS" is used to indicate a 16-bit binary address. In the case of segmented mode instructions the address fields are designated as "SEGMENT" and "OFFSET." In some RA and BA addressing mode instructions "DISPLACEMENT" indicates a binary field containing a displacement supplied by the assembler from the label (address), displacement, or index specified in the assembler language syntax. Other fields specify condition codes ("CC"), number of or location of bits or shift positions ("b" or "n"), or flags ("C", "Z", "S", "PV", "V", "N", etc).

These and other notations are defined in following sections.

Note from Figure 5.5.1 that bit eight of the opcode distinguishes between a word instruction (bit 8 = 1) or a byte instruction (bit 8 = 0)

Appendix E gives a complete opcode map for the AmZ8001 and AmZ8002 CPUs.

5.2.3 Addressing Mode Encoding

Section 4.3 and Figure 3.6.1 discuss address mode encoding within the instruction opcodes. Pertinent figures are included in the summary material of Section 5.5.

The instruction encodes the addressing modes by using the mode bits (15 and 14), bits 13 and 12, and the register field (bit 7 through bit 4). The specification of general purpose register (pair) zero, R0 or RR0, in the register field of the instruction is used to distinguish between combinations of the addressing modes.

If the instruction mode bits 15 and 14 are "00" and bits 13 and 12 are not "11", then specifying R0 (or RR0) in the register field (designation = "0000") denotes IM mode, and any other register designation denotes IR mode. (There are exceptions to this.) Likewise, if the mode bits are "00" and bits 13 and 12 are "11," then specifying R0 denotes RA mode, and any other register designation denotes BA mode.

If the mode bits are "01," specifying R0 denotes DA mode, and any other register specification designates X or BX mode. Mode bits "10" designate R addressing mode, and "11" designate special instructions such as the short one-word instructions.

5.2.4 Use of Register R0 and RR0

From the above discussion it is seen that R0 (or RR0) cannot be designated in the instruction register field for the X, BA, and BX addressing modes. That is, general purpose register (pair) zero cannot be used as an index register in X mode or as a base address register in BA or BX modes. In the IR mode some instructions allow the designation of R0 while other instructions do not. The instruction pages following accurately define, in the instruction register fields and descriptions, whether R0 or RR0 can be designated.

Another restriction of using R0 or RR0 is as a stack pointer when using the PUSH and POP instructions. Register (pair) zero cannot be designated as a stack pointer.

To summarize R0 (or RR0) can be designated just as any other register (pair) in the instruction register field with some restric-

tions. Whether R0 can be designated in a particular addressing mode or form of an instruction is shown on the detailed instruction pages. The general rules for using R0 (and RR0) are listed here.

- X Mode: R0 cannot be designated as the index register in indexed (X) addressing mode.
- BA, BX Modes: R0 cannot be designated as the base address register in base address (BA) or base indexed (BX) addressing modes. Note that R0 or RR0 can be designated as the index register in the BX mode.
- IR Mode: R0 cannot be designated as the indirect register in the IR mode if that instruction has the immediate form (IM mode) also available. Two other considerations apply for the IR mode:
- a. If no IM mode of the instruction is available, R0 (or RR0) can be designated as the indirect register in the IR mode. An example of this is the INCREMENT Word Instruction, INC, for both segmented and nonsegmented versions of the IR addressing mode.
- b. In other cases where the IM mode of the instruction is not available, the designation of R0 (or RR0) is not allowed in that it is used to distinguish a different version or instruction. An example is the SET Bit in Word instruction, SET, which has no IM mode. Specifying R0 or RR0 in IR mode of the static form of the instruction is not allowed because specifying "0000" in the register field is the opcode for the dynamic form of the instruction (R mode).
- Stack Pointer: R0 or RR0 cannot be designated as stack pointers in the PUSH and POP instructions.

5.3 ASSEMBLER LANGUAGE SYNTAX

Each form of an instruction (including byte, word, and long word, segmented and nonsegmented versions, and each available addressing mode) is shown with its corresponding assembler language statement. The statements follow the general instruction notation used by the MACRO8000 assembler. For additional information refer to the MACRO8000 Assembler User's Manual.

The assembler syntax is given for each mnemonic at the top center of each page just below the name of the instruction. This statement is a single generic form that covers all variations of the instruction on that page. Though generic, the statement shows specific operands wherever possible; these are indicated by upper case letters. Lower case letters are used to denote variables in the statement for which suitable values are to be substituted. One example of this is "Rs," denoting a word register (R0, R1, ..., or R15) required by the addressing mode as a source register. Another example is the use of "dst" ("src"), indicating a general destination (source) which type is determined by the addressing mode.

The assembler syntax is also given for each specific version of the instruction and is shown directly above its binary machine code representation. Because these are specific forms, the general "dst" and "src" operands are substituted by their specific value for that addressing mode.

Some variables are shown in upper case. These include labels, displacements, lists, and binary and integer values. They are listed in the following section entitled Notation key.

5.4 ASSEMBLER EXCEPTIONS

MACRO8000 does not allow designation of R0 or RR0 as an indirect register in the IR addressing mode. Some instructions do allow the designation of R0 (or RR0) in the AmZ8001 and AmZ8002. These are noted in the detailed instruction pages.

The assembler statements are terminated with a semicolon (;). These have not been shown.

The MACRO8000 does not use the LDA or LDAR mnemonic to implement the LOAD Address instruction. This instruction is implemented using the LD mnemonic with the address specified as an immediate operand address constant. This may generate an LD or LDA instruction object code, depending on the addressing mode and the version of the assembler being used. To specify an address constant to be used as an immediate operand, place a "\tilde{\top}" character in front of the operand. (Refer to comments made in Section 5.6.8 regarding use of the circumflex character, "\tilde{\top}".")

The LDB load byte into a register with an immediate value instruction has two opcodes which will perform the desired operation. MACRO8000 generates the faster and shorter version; other assemblers may support both forms.

The instructions listed below have a comment in the Assembler Notation section as follows: "A LAB or D which results in a displacement outside the allowable range produces an assembler error." Because the displacement range of these instructions in the RA and BA addressing modes is +32,767 to -32,768, the MACRO8000 assembler can compute the correct displacement if outside the indicated range. It does this by generating a module 65,536 two's complement displacement. For example, a LAB indicating a displacement of -65,520 will produce a positive displacement of +16 by wrapping around.

No error occurs in these cases. Instructions which have displacements assembled this way include:

LD. LDR word register into memory

LD, LDR word into register

LDB, LDRB byte register into memory

LDB, LDRB byte into register a most part sollowing ensures

LDL, LDRL long word register into memory

LDL, LDRL long word into register

LDA, LDAR address into register

5.5 SUMMARY OF ARCHITECTURAL DETAILS

The following figures and tables are a reproduction of previous information provided here for easy reference in understanding the instruction set details.



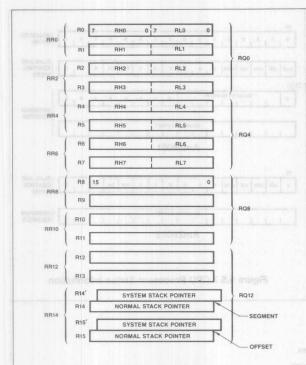


Figure 5.5.1 AmZ8001 General Registers



Register Designator	Byte Mode	Word Mode	Word Mode	Quadruple Word Mode
0000	RH0	R0	RR0	RQ0
0001	RH1	R1	_	
0010	RH2	R2	RR2	il out
0011	RH3	R3		-
0100	RH4	R4	RR4	RQ4
0101	RH5	R5	-	_
0110	RH6	R6	RR6	1074
0111	RH7	R7	- 1	
1000	RL0	R8	RR8	RQ8
1001	RL1	R9	0 - 10	12
1010	RL2	R10	RR10	59
1011	RL3	R11	-	
1100	RL4	R12	RR12	RQ12
1101	RL5	R13	ar - or	0.25
1110	RL6	R14	RR14	-
1111	RL7	R15	6 L	108

(-Reserved)

Notes

All general purpose registers can be used as accumulators. However, R0 in the AmZ8002 (and RR0 in the AmZ8001) cannot be used as an index register or memory pointer. Refer to the section on Address Modes (3.6) Section 5.2.4.

The highest order general-purpose registers are used as implied stack pointers. For a description of this refer to the section entitled Stack Pointers (2.3.4).

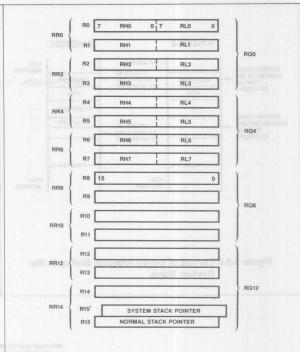
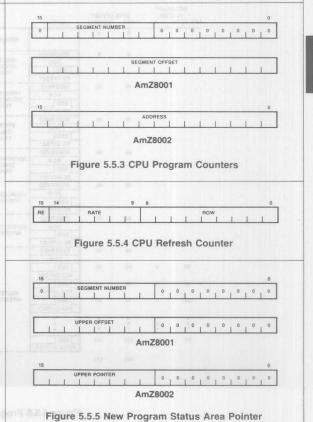


Figure 5.5.2 AmZ8002 General Registers



5-3

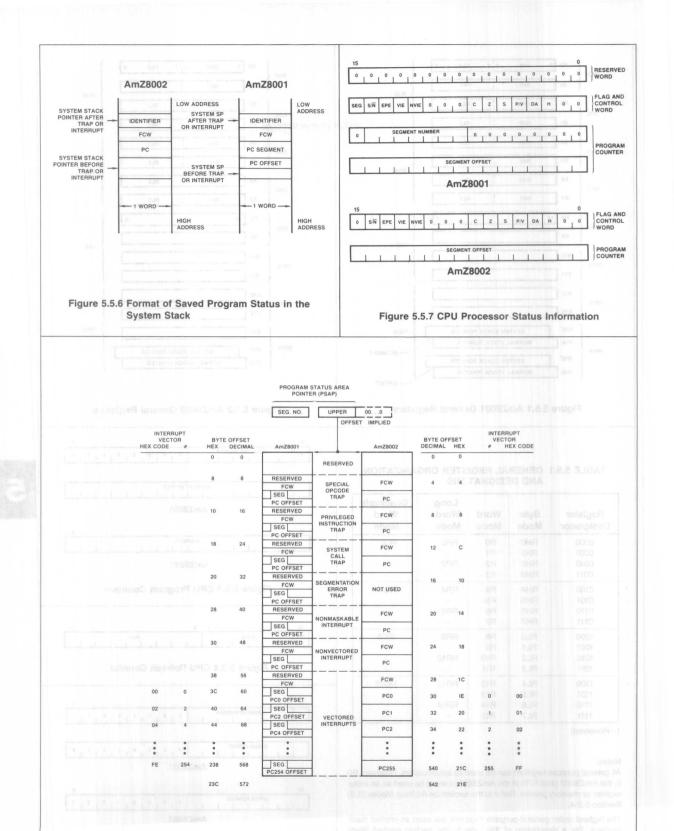


Figure 5.5.8 Program Status Area

E OPCODE W/B REG REG	MAIN FIELDS OF INSTRUCTION — ONE OR TWO WORDS	7 6 5 4 3 2 1 0	BITS IN A BYTE
)		BITS IN A WORD
SEGMENT	ADDRESS OR DISPLACEMENT	ADDRESS n	BYTE
OFFSET	FIELD - ZERO, ONE. OR TWO WORDS	ADDRESS n (EVEN) ADDRESS n + 1 (ODD) UPPER BYTE LOWER BYTE	WORD
MS WORD		ADDRESS n ADDRESS n + 1 UPPER WORD/UPPER BYTE	84
MS WORD	IMMEDIATE OPERAND FIELD-ZERO, ONE, OR TWO WORDS	ADDRESS n + 2 ADDRESS n + 3 LOWER WORDLOWER BYTE	LONG WORD

	Address Mode			
Bits 15, 14	Bits 13, 12	Bits 7, 6, 5, 4	Mode	
00	Not 11	0	IM	
00	Not 11	Not 0	IR	
00	11	0	RA	
00	11	Not 0	BA	
01	X	0	DA	
01	Not 11	Not 0	X	
BI 01 8889	10a 9118 28	Not 0	BX	
10	Total X month	X X	R	
11	X	X	Special*	

*Used for short one-word instructions.

Word/Byte

1	and the second	Burker L.	
	Bit 8	Mode	
	0	Byte	
	1	Word	

Displacement -	16-Bit Word
Nonsegmented Address	16-Bit Word
Segmented Short Offset	0 + 7-Bit Segment + 8-Bit Offset
Segmented Long Offset	First Word: 1 + 7-Bit Segment + 8-Bit Unused Second Word: 16-Bit Offset

Immediate Data, When Present

Byte	Same Byte in Both Halves of Word
Word	16-Bit Word
 Long-Word	First Word: Bits 16:31 of Operand Second Word: Bits 0:15 of Operand

TABLE 5.3 CC-FIELD DECODING

CC Field	Assembler Notation	Meaning	Flag Settings for CC True
1110	NZ	Not Zero	Z = 0
0110	ZR	Zero	Z = 1
n 11110 en	NC	No Carry	C = 0
0111	CY	Carry	C = 1
1100	PO	Parity Odd	P/V = 0
0100	PE	Parity Even	P/V = 1
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
1110	NE	Not Equal	Z = 0
0110	EQ	Equal	Z = 1
1100	NOV	Overflow is Reset	P/V = 0
0100	OV	Overflow is Set	P/V = 1

1001	GE	Greater Than or Equal	S ⊕ P/V = 0
0001	LT	Less Than	S ⊕ P/V = 1
1010	GT	Greater Than	$Z + (S \oplus P/V) = 0$
0010	LE	Less Than or Equal	$Z + (S \oplus P/V) = 1$

1111	LGE	Logical Greater Than or Equal	C = 0
0111	LLT	Logical Less Than	C = 1
1011	LGT	Logical Greater Than	$(C = 0) \cdot (Z = 0) = 1$
0011	LLE	Logical Less Than or Equal	C + Z = 1
1000		Unconditional (Always True)	

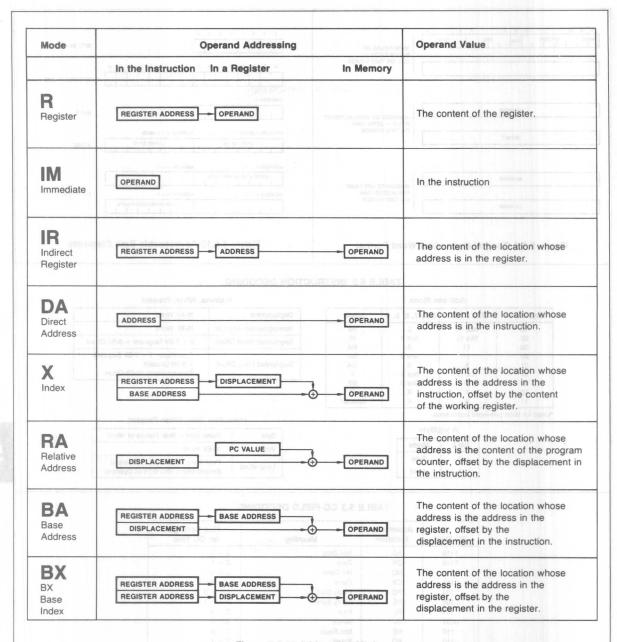


Figure 5.5.11 Addressing Modes

5.6 NOTATION KEY

Abbreviations and notations used on the detailed instruction pages are included in this section.

5.6.1 Appended Information

Refer to the appendices for additional information:

riorer to the up	portation to administration and the second
Appendix A	AmZ8000 Instruction Set: By Logical Group
Appendix B	AmZ8000 Instruction Set: Numeric Listing by Opcode
Appendix C	AmZ8000 Instruction Set: Alphabetic Listing By Mnemonic
Appendix D	AmZ8000 Instruction Set: Topical Index
Appendix E	AmZ8000 Instruction Set: Opcode Map
Appendix F	Executive Module Sample Code
Appendix G	ASCII Character Set
Appendix H	Powers of 2 and 16
Appendix I	Hexadecimal and Decimal Integer Conversion Table

5.6.2 Addressing Mode and Segmentation Notation

Notation used to designate the addressing modes is shown and includes examples of assembler syntax operands.

Notation	Addressing Mode	Assembler Syntax Example	
R	Register	R7 879 88	
IM	Immediate 51 255	4 CONSTANT ↑LABEL	
IR	Indirect Register	R7↑	
DA	Direct Address	LABEL #4D8B LABEL + DISPLACEMENT	
X	Index	LABEL (R7) #4D8B↑ (R7)	
RA	Relative Address	LAB \$ + #100)	
BA	Base Address	R7↑(DISPLACEMENT) R7↑(#100)	
BX	Base Index	R7↑(R1)	
PR	Port Register	R7	
PA	Port Address	4 CONSTANT	

The segmentation version notation is:

NS - nonsegmented

S - segmented

SSO - segmented short offset

SLO - segmented long offset

5.6.3 Source and Destination Notation

Operand sources and destinations are indicated in the general instruction assembler statement at the top of the page. Non-specific forms are indicated by lower case "src" and "dst." This implies that the addressing mode determines the specific source or destination.

Specific sources and destinations are listed for each form and addressing mode of the instruction and, if all versions on the page are identical, at the top of the page. The specific source is designated by a lower case "s" suffix behind a register notation. The specific destination is designated by a lower case "d" suffix behind a register notation. Examples of these are "Rs" and "RRd"

5.6.4 Register Notation

Notation	Meaning	Range/Examples	Comments
*R	word register	R0, R1,, R15	16 bits
*Rb	byte register	RL0, RH0, RL1,, RH7	8 bits
*RR	register pair	RR0, RR2,, RR14	32 bits
*RQ	register quad	RQ0, RQ4, RQ8, RQ12	64 bits
Rx	index register	as, R7	word register only
Rc	counter register	as, R10	word register only
Rbc	byte counter register	as, RH7	byte register only
Rp	port register	as, R14	word register only

*-Appended with a source or destination indication, "s" or "d."

Examples are Rs, Rbd, RRs, RQd, etc.

5.6.5 Operand Notation 1980 bills esteed notified 1.8.8

In addition to register operand designations, notations used for operands and other values are listed here both for the assembler language statements and the instruction opcodes. Upper case is used for statement operands and correspond to an instruction field value, typically indicated in lower case. The assembler assembles the statement operand into the appropriate binary code. A statement about assembler notation is given on instruction pages where appropriate and includes information on range and relationship between assembler and opcode values.

Notation	Description		
b, B	bit number or number of bit positions		
c, C	count or carry		
CC	condition code		
CR	control register		
d, D	signed, 2's complement displacement		
h	hex integer; 01, 1,, F		
IMITOGI	immediate operand (IM = word, IMb = byte, $IM \ell = long$ -word, IMd = digit)		
LIST	list of optional (all, any or none) values		
n, N	decimal integer		

5.6.6 Address and Label Notation

The general assembler statements and operations use "addr" to denote a memory address. These addresses are generated by the assembler depending on the addressing mode and assembler syntax, such as labels, displacements, and indexes. Addresses also take the form #hh (segment) or #hhhh (offset). An example is #4D8B, the # symbol designating a hex integer value.

Notation	Description
ADDRESS	Denotes a 16-bit binary address field in the instruction generated by the assembler for nonsegmented versions of DA and X mode instructions.
DISPLACEMENT A signed or unsigned 2's complement integer used in the RA and BA addressing more relative instructions such as CALL Subroutine (CALR), JUMP Conditional (JR), an instructions such as LDR. The instruction description and assembler notation define the displacement is added or subtracted and what the range is.	
LAB An assembler syntax label designating the relative address (RA) mode. The assemthis label to generate the instruction displacement relative to the updated PC.	
LABEL An absolute address in memory specified to the assembler either as a label or a address of the form #hhhh.	
LABSSO A label designating a segmented short offset (SSO) form of address. This is con 7-bit segment address and the lowest byte of a 16-bit offset address. It is used to SSO version of DA and X mode instructions.	
OFFSET	A field used to specify the offset address (16 bits for SLO or 8 bits for SSO) in a segmented form of DA or X mode instructions.
PORT An assembler syntax used to denote a port address (a 1-bit field) in a port address (PA of the (special) I/O instructions.	
SEGMENT A field used to specify the 7-bit segment address in segmented (SSO or SLO) for and X mode instructions.	

Note that the assembler uses segment directives, not instruction statements, to specify segment address to be used during assembly. Refer to the assembler reference manual.

5.6.7 Condition Codes and Other Notations

The condition codes use a notation defined in Table. Also shown are the CPU flag settings. In addition to these, other notations are used:

Notation	Meaning	Instruction Example
CC	condition code	JR
CR	control register	LDCTL
FCW	flag and control word	LDCTL
FLAGS	flag byte of FCW	LDCTLB
N	non-vectored interrupt opcode bit	El
NSPOFF	normal stack pointer offset	LDCTL
NSPSEG	normal stack pointer segment	LDCTL
NVIE	non-vectored interrupt flag	EI Ga
PSAPOFF	POFF NPSAP upper offset	
PSAPSEG	NPSAP segment	LDCTL
REFRESH	refresh register	LDCTL
SGN	sign bit	COMFIG
V	vectored interrupt opcode bit	EI
VIE	vectored interrupt flag	EI

5.6.8 Special Character Notation

The operations described on each instruction page use the following convention with respect to special characters:

Symbol	Syntax	Description	olte
-	"is replaced by"		P
<>	denotes bit fi	eld, range or position	
()	denotes "cor	ntents of"	
:	indicates low	est and highest bit posit	ions
;	delimiter for	list of optional entries	
+	add		
REDAU9	subtract		
X	times		
÷ or /	divide		
V	logical OR		
٨	logical AND		
+	logical EXCL	USIVE OR	
_	logical compl	ement	
\rightarrow	denotes direc	ction of rotation or shift	
\longleftrightarrow	exchange		
=	equals		
≠	not equals		

5

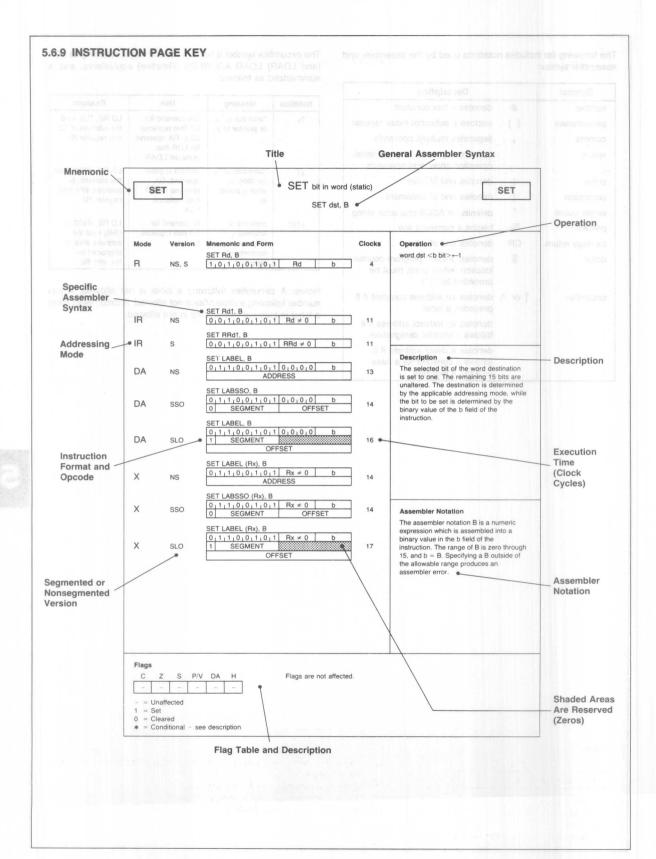
The following list includes notations used by the assembler and assembler syntax:

Symbol		Description		
number	#	denotes a hex constant		
parentheses	()	enclose a subscript index register		
comma	,	separates multiple operands		
space		element separator between label, operation code, and operands		
colon	-:	denotes end of label		
semicolon	;	denotes end of statement		
single quote	,	delimits an ASCII character string		
percent	%	begins a comment line		
carriage return	CR	denotes end of a line		
dollar	\$	denotes present program counter location (when used, must be preceded by "↑")		
circumflex	↑ or ∧	denotes an address constant if it precedes a label		
		denotes an indirect address if it follows a register designation		
	resident out-state	denotes a direct address if it follows a constant (or a base address if indexed)		

The circumflex symbol is used in several ways, including as LDA (and LDAR) LOAD ADDRESS (Relative) equivalents, and is summarized as follows:

Notation	Meaning	Use	Example	
1x	"address of" χ , or pointer to χ	DA operand for LD that replaces LDA; RA operand for LDR that replaces LDAR	LD R2, †L3; load the address of L3 into register R2	
x† /	"contents at" χ location, or what χ points to	indirect register operand; DA operand for LD that replaces LDA	LD R2, #F4†; load the contents at address #F4 into register R2	
χ ↑ (r)	contents at address χ displaced by contents of register r	X operand for LD that replaces LDA	LD R2, #4320↑ (R4); load the address #4320, displaced by R4, into R2	

Notes: A circumflex following a label is not allowed. A hex number following a circumflex is not allowed. A circumflex before a label in indexed instructions is not allowed.



ADC

ADD words with carry

ADC

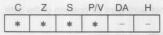
ADC Rd, Rs

Mode	Version	Mnemonic and Form			Clocks	Operation
		ADC Rd, Rs				Rd<0:15>←Rs<0:15>+Rd<0:15>+C
R	NS, S	1,0,1,1,0,1,0,1	Rs	Rd	5	A SELECTION BANK A

Description

The contents of the general-purpose registers designated by the Rs (source) and Rd (destination) fields of the instruction are added together along with the carry flag to obtain the result. The 16-bit result is loaded into the destination register, whose original contents are lost. The contents of the source are not altered.

Flags



- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

- C: Set to 1 if there is carry from the most significant bit position of the word. Reset otherwise.
- Z: Set to 1 if result is zero. Reset otherwise.
- S: Set to 1 if result is negative. Reset otherwise.
- P/V: Set to 1 on arithmetic overflow. Reset otherwise.

5

ADCB

ADD byte with carry

ADCB Rbd, Rbs

ADCB

Mode Version 3 = <81 0 pb/4 + <8 R NS, S	miletino ino ana i orini	Rbs Rbd	Clocks	Operation Rbd<0:7>←Rbs<0:7>+Rbd<0:7>+C
				Description The contents of the general-purpose byte registers designated by the Rbs (source) and Rbd (destination) fields of the instruction are added together along with the carry flag to obtain the result. The 8-bit result is loaded into the destination register, whose original contents are lost. The contents of the source are not altered.

=	Unarrected	
=	Set	
_	Classad	

S P/V DA H

- 0

Flags

- = Conditional see description
- C: Set to 1 if there is a carry from the most significant bit position of the byte. Reset otherwise.
- Z: Set to 1 if result is zero. Reset otherwise.
 S: Set to 1 if result is negative. Reset otherwise.
 - P/V: Set to 1 on arithmetic overflow. Reset otherwise.
 - DA: Reset always.
 - Set to 1 on carry from the least significant digit of result. Reset otherwise.

ADD word to register

ADD

ADD Rd, src

Mode	Version	Mnemonic and Form	Clocks	Operation		
		ADD Rd, Rs		Rd<0:15>←src<0:1	5>+Rd<0:15>	>
7	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 Rs Rd	4	[10,0,0,0,0,0,0]		
		ADD Rd, IM		ADDB Ribd, 1Mb		
М	NS, S	0,0,0,0,0,0,0,1 0,0,0,0 Rd	7 7 10 11	10,0,0,0,0,0,0		
		OPERAND		OPERAND		
		ADD Rd, Rs↑		ADDB RbJ, Ref		
R	NS	$0_10_10_10_10_10_11$ Rs $\neq 0$ Rd	0 + 7 1	10,0,0,0,0,0,0,0		
		ADD Rd, RRs↑		ADDB Red, RRef		
R	S	$0_10_10_10_10_10_10_11$ RRs $\neq 0$ Rd	7 7	10.0.0.0.0.0.0.0	2	- 5
		ADD Rd, LABEL		Description		
DA	NS	0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd	9 0	Source operand and		
210-	8 piti bing left	ADDRESS	Besh	words are added tog		
		ADD Rd, LABSSO		contents of the source		
DA	SSO	0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd	10	and the original cont		
		0 SEGMENT OFFSET		destination are lost. determined by the a		coina
		ADD Rd, LABEL		mode and the destin		
DA	di en video	0,1,0,0,0,0,1 0,0,0,0 Rd 1 SEGMENT	12	general-purpose reg		d by
DA	SLO	1 SEGMENT OFFSET	12	the Rd field of the in	struction.	
		ADD D4 LAREL (Dv)				
		ADD Rd, LABEL (Rx) 0 1 1 0 1 0 1 0 1 0 1 1 Rx ≠ 0 Rd		ADDS RED LABEL (R		
X	NS	ADDRESS	10	IGA		
		ADD Rd, LABSSO (Rx)		ADDS Rbd. LABSSO		
X	SSO	$0_1 1_1 0_1 0_1 0_1 0_1 0_1 1$ Rx $\neq 0$ Rd	10	10,0,0,0,0,1,0		
^	330	0 SEGMENT OFFSET	10	O SEGMENT		
		ADD Rd, LABEL (Rx)		AODE REG. LABEL (R		
		$0_{1}1_{1}0_{1}0_{1}0_{1}0_{1}0_{1}1$ Rx $\neq 0$ Rd		0.0101010110		
X	SLO	1 SEGMENT OFFSET	13	THEMBER		

Flags



- = Unaffected
- 1 = Set
- * = Conditional see description
- C: Set to 1 if there is a carry from the most significant bit position of the word. Reset otherwise.
- Z: Set to 1 if result is zero. Reset otherwise.
 S: Set to 1 if result is negative. Reset otherwise.
- P/V: Set to 1 on arithmetic overflow. Reset otherwise.

ADDB

ADD byte to register

ADDB Rbd, src

ADDB

Mode	Version	Mnemonic and Form Clock	s Ope	ration		
		ADDB Rbd, Rbs	Rbd-	<0:7>←src<0:7	'>+Rbd<0:7>	
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 Rbs Rbd 4	10,0			
		ADDB Rbd, IMb				
IM	NS, S	010101010101010101010 Rbd	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			
IIVI	NS, S	7 OPERAND 0 7 OPERAND 0	1AR 190			
		ADDB Rbd, Rs↑				
IR	NS	$0_10_10_10_10_10_10$ Rs $\neq 0$ Rbd $0 = 7$	1 1000			
		ADDB Rbd, RRs↑				
IR	S	$0_10_10_10_10_10_10_10$ RRs $\neq 0$ Rbd 7	8 0,0	0,0,0,0,0	2	
		ADDB Rbd, LABEL	Des	cription		
DA	NS	0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 Rbd	Sou	ce operand and	destination ope	erand
16.91		put belon ins abrovADDRESS	byte	s are added toge		
nd ar	ether and the he destine for	g f bebby to sprovADDRESS	byte: resu	s are added toge It is loaded into t	he destination.	The
en Tur het	emer and the for he desire for each of the	ADDRESS ADDB Rbd, LABSSO	byte resu cont	s are added toge It is loaded into the ents of the source	he destination. e are not altere	The ed and
nd ar	ether and the he destine for	ADDB Rbd, LABSSO	byte resu cont the c lost.	s are added toge It is loaded into the ents of the source original contents The source is de	he destination. ee are not altere of the destinati etermined by th	The ed and on are
en Turan het	emer and the for he desire for each of the	ADDRESS ADDB Rbd, LABSSO 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	byte resu cont the c lost. appl	s are added toge It is loaded into the ents of the source original contents The source is de icable addressing	he destination. ee are not altere of the destinati etermined by the g mode and the	The ed and on are
en Turan het	emer and the for he desire for each of the	ADDRESS ADDB Rbd, LABSSO 0 1 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0	byte resu cont the colost. appl dest	s are added toge It is loaded into the ents of the source original contents The source is de icable addressing ination is always	he destination. be are not altered of the destinative termined by the g mode and the a general-purp	The ed and on ar le e e e e e e e e e e e e e e e e e e
on Turan hat	emer and the for he desire for each of the	ADDRESS ADDB Rbd, LABSSO 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	byte resu cont the c lost. appl dest byte	s are added toge It is loaded into the ents of the source original contents The source is de icable addressing	he destination. be are not altered of the destinative termined by the g mode and the a general-purp	The ed and on ar le e e e e e e e e e e e e e e e e e e
DA	SSO and solve an	ADDRESS ADDB Rbd, LABSSO 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	byte resu cont the c lost. appl dest byte	s are added toge It is loaded into the ents of the source original contents The source is de icable addressing ination is always register designa	he destination. be are not altered of the destinative termined by the g mode and the a general-purp	The ed and on armie
DA posses	SSO and solve an	ADDRESS ADDB Rbd, LABSSO 0	byte resu cont the colost. appl dest byte of the	s are added toge It is loaded into the ents of the source original contents The source is de icable addressing ination is always register designa	he destination. be are not altered of the destinative termined by the g mode and the a general-purp	The ed and on are e
DA DA	SSO and SLO	ADDRESS ADDB Rbd, LABSSO 0 1 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0	byte resu cont the c lost. appl dest byte of th	s are added toge It is loaded into the ents of the source original contents The source is de icable addressing ination is always register designa e instruction.	he destination. The are not altered of the destination of the destinat	The ed and on armie
DA DA	SSO and solve an	ADDRESS ADDB Rbd, LABSSO 0	byte resu cont the colost. appl dest byte of the	s are added toge It is loaded into the ents of the source original contents The source is de icable addressing ination is always register designa e instruction.	he destination. be are not altered of the destinative termined by the g mode and the a general-purp	The ed and on ar le e e e e e e e e e e e e e e e e e e
DA posses	SSO and SLO	ADDRESS ADDB Rbd, LABSSO 0 1 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0	byte resu cont the c lost. appl dest byte of th	s are added toge It is loaded into the ents of the source original contents The source is de icable addressing ination is always register designa e instruction.	he destination. The are not altered of the destination of the destinat	The ed and on armie
DA DA	SSO SLO NS	ADDRESS ADDB Rbd, LABSSO 0	byte resu cont the c lost. appl dest byte of th	s are added toge It is loaded into the ents of the source original contents. The source is de icable addressing ination is always register designa e instruction.	he destination. e are not altere of the destinati etermined by th g mode and the a general-purp ted by the Rbd	The ed and on armie
DA DA	SSO and SLO	ADDRESS ADDB Rbd, LABSSO 0	byte resu cont the c lost. appl dest byte of th	s are added toge It is loaded into the ents of the source original contents. The source is de icable addressing ination is always register designa e instruction.	he destination. The are not altered of the destination of the destinat	The ed and on are e
DA DA	SSO SLO NS	ADDRESS ADDB Rbd, LABSSO 0	byte resu cont the c lost. appl dest byte of th	s are added toge It is loaded into the ents of the source original contents. The source is de icable addressing ination is always register designa e instruction.	he destination. e are not altere of the destinati etermined by th g mode and the a general-purp ted by the Rbd	The ed and on are e
DA DA	SSO SLO NS	ADDRESS ADDB Rbd, LABSSO 0	byte resu cont the cont	s are added toge It is loaded into the ents of the source original contents. The source is de icable addressing ination is always register designa e instruction.	he destination. e are not altere of the destinati etermined by th g mode and the a general-purp ted by the Rbd	The ed and on are e



* *

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- Z S P/V DA H C: Set to 1 if there is a carry from the most significant bit position of the byte. Reset otherwise.
 - Set to 1 if result is zero. Reset otherwise.
 - S: Set to 1 if result is negative. Reset otherwise. P/V: Set to 1 on arithmetic overflow. Reset otherwise.
 - DA: Always reset.
 - H: Set to 1 if there is a carry from the least significant digit. Reset otherwise.

ADDL

ADD long word to register

ADDL

4001	001	
ADDL	нна,	src

				T	
Mode	Version	Mnemonic and Form	Clocks	Operation	
		ADDL RRd, RRs		RRd<0:31> + src<0:31> + RRd<0	
R	NS, S	1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 RRs RRd	8	1.1.7.0.0.0.0.1 2 au	
		ADDL RRd, IMI			
		0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 RRd		AND Rd, RM	
IM	NS, S	31 OPERAND 16 15 OPERAND 0	14	NS. S 0,0,0,0,0,0,11,1	
		15 OPERAND 0			
		ADDL RRd, Rs↑		AND Rd, Rst	
IR	NS	$0_10_10_11_10_11_11_10$ Rs $\neq 0$ RRd	14	NS [0,0,0,0,0,1,1,1	
		ADDL RRd, RRs↑		AND Rd. RRst	
IR	S	$0_10_10_11_10_11_11_0$ RRs $\neq 0$ RRd	14	11.11.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0	
		ADDL RRd, LABEL		Description	
		0,1,0,1,0,1,1,0,0,0,0,0 BBd		Source operand and destination op	
	NS	ADDRESS	15	long words are added together and result is loaded into the destination	
		ADDL RRd, LABSSO		contents of the source are not alter	
birth c				the original contents of the destina	
DA	SSO	0 SEGMENT OFFSET	16	lost. The source is determined by tapplicable addressing mode and the	
		ADDL RRd, LABEL		destination is always a general-pur	
		0 ₁ 1 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 RRd		register pair designated by the RR	d field
DA	SLO	1 SEGMENT	18	of the instruction.	
		OFFSET		110	
		ADDL RRd, LABEL (Rx)		AND Ed. LABEL (Rx)	
X		$0_11_10_11_10_11_11_0$ Rx $\neq 0$ RRd	1 Rox ≠ D	1 1 1 1 0 10 10 11 10 204	
X	NS	ADDRESS	16	GA	
		ADDL RRd, LABSSO (Rx)		AND Rd. LABSSO (Rd	
V		$0_1 1_1 0_1 1_1 0_1 1_1 1_1 0$ Rx $\neq 0$ RRd		1,1,1,0,0,0,1,0	
X	SSO	0 SEGMENT OFFSET	16	TABIMASE TO COS	
		ADDL RRd, LABEL (Rx)		IXAL IBEAL BR GHA	
		$0_11_10_11_10_11_11_0$ Rx $\neq 0$ RRd		1,1,1,0,0,0,1,6	
X	SLO	1 SEGMENT	19	ELO 1 SEGMENT	
		OFFSET			

Flags

С	Z	S	P/V	DA	H
a c	*	*	*	-	el+ris

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Set to 1 if there is a carry from the most significant bit position of the long word. Reset otherwise.

 - Z: Set to 1 if result is zero. Reset otherwise.
 S: Set to 1 if result is negative. Reset otherwise.
 P/V: Set to 1 on arithmetic overflow. Reset otherwise.

AND

AND word with register

AND Rd, src

AND

Mode	Version		Clocks	Operation
R	NS, S	AND Rd, Rs 1 0 0 0 0 1 1 1 Rs Rd	8A 4	Rd<0:15>←Rd<0:15> ∧ src<0:15>
IM	NS, S	AND Rd, IM 0 0 0 0 0 0 0 10 1 1 1 1 0 0 0 0 0 Rd OPERAND	7	36.) 655 JOCA 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
IR	NS	AND Rd, Rs↑ 0,0,0,0,0,1,1,1,1 Rs ≠ 0 Rd	7	ADDLERBALING
IR	S	AND Rd, RRs↑ 0 0 0 0 0 1 1 1 RRs ≠ 0 Rd	7	1-82 - 800 - 400A
DA	NS	AND Rd, LABEL 0 1 0 0 0 1 1 1 0 0	9	Description A logical AND operation is performed between the corresponding bits of the
one bar	SSO	AND Rd, LABSSO 0 1 1 0 1 0 1 1 1 1 0 1 0 1 0 Rd 0 SEGMENT OFFSET	10	source and destination words. The source operand is determined by the applicable addressing mode, while the destination operand is always a general-
	SLO	AND Rd, LABEL 0 1 1 0 1 0 1 1 1 1 0 1 0 1 0 1 0 Rd 1 SEGMENT OFFSET	12	purpose word register, designated by the Rd field of the instruction. The result of the operation is loaded into the destination, whose original contents are lost. The source contents are not altered.
X	NS	AND Rd, LABEL (Rx) 0 1 1 0 1 0 1 0 1 1 1 1 1 Rx ≠ 0 Rd ADDRESS	10	ADDL RRd. LABEL 9 0.01.01.01.01.01
×	SSO	AND Rd, LABSSO (Rx) 0 1 1 0 1 0 1 1 1 1 1 1 1 1 Rx ≠ 0 Rd 0 SEGMENT OFFSET	10	ADDL HRB. LABSSO 0.11.01.0 0.11.1.1 0. SECMENT
	SLO	AND Rd, LABEL (Rx) 0 1 0 0 1 1 1 Rx ≠ 0	13	ADDLARD LABEL O

Flags

C	Z	S	P/V	DA	Н
-	*	*	-	-	-

- Z: Set to 1 if result is zero. Reset otherwise.
 - S: Set to 1 if result is negative. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

AND byte with register

ANDB

ANDB Rbd, src

Mode	Version	Mnemonic and Form	Clocks
		ANDB Rbd, Rbs	
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 Rbs Rbd	4
		ANDB Rbd, IMb	
IM	NS, S	0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd	7
	1,0,0	7 OPERAND 0 7 OPERAND	0 '
_		ANDB Rbd, Rs↑	_
IR	NS	$0_10_10_10_10_11_10$ Rs $\neq 0$ Rbd	7
		ANDB Rbd, RRs↑	
IR	S	$0_10_10_10_10_11_11_10$ RRs $\neq 0$ Rbd	7
		ANDB Rbd, LABEL	
DA	NS	0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd] 9
	145 D D D D D D D D D D D D D D D D D D D	ADDRESS	
		ANDB Rbd, LABSSO	
DA	SSO	0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd	10
lo sile	Fasal treation	0 SEGMENT OFFSET	
		ANDB Rbd, LABEL	_
DA	01.0	0 1 0 0 0 1 1 1 0 0 0 0 0 0 Rbd	
DA	SLO	1 SEGMENT OFFSET	<u>×</u> 12
		AUDD DI LI ADEL (D.)	
		ANDB Rbd, LABEL (Rx) $0_11_10_10_10_11_11_10 \text{ Rx} \neq 0 \text{ Rbd}$	7
X	NS	ADDRESS	10
		ANDB Rbd, LABSSO (Rx)	
V		$0_1 1_1 0_1 0_1 0_1 1_1 1_1 0$ Rx $\neq 0$ Rbd	٦
X	SSO	0 SEGMENT OFFSET	10
		ANDB Rbd, LABEL (Rx)	
		$0_11_10_10_10_11_11_0$ Rx $\neq 0$ Rbd	
X	SLO	1 SEGMENT	₩ 13
		OFFSET	

Description

A logical AND operation is performed between the corresponding bits of the source and destination bytes. The source operand is determined by the applicable addressing mode, while the destination operand is always a general-purpose byte register, designated by the Rbd field of the instruction. The result of the operation is loaded into the destination, whose original contents are lost. The source contents are not altered.

Operation
Rbd<0:7>←Rbd<0:7>∧ src<0:7>

Flags

С	Z	S	P/V	DA	Н
-	*	*	*	-	-

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 if parity of result is even. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

BIT

BIT test in a word (dynamic)

BIT Rd, Rs

BIT

Mode	Version	Mnemonic and Form BIT Rd, Rs			Clocks	Operation Z flag←Rd <bit in="" rs(0:3)="" specified=""></bit>	
R	NS, S	0,0,1,0,0,1,1,1	0,0,0,0	Rs	10		
					7 026		
					0 4 3 8		
					Q N a SA	ANDS Rod, RRs1	7
					10,0,0,01 RESS	Description The selected bit of the word destinative register is tested and the Z flag is affected. The destination word operations are considered.	nd is
					0,0,0,0 330	the general-purpose register designate by the Rd field of the instruction. The be tested is determined from a binary decode of the least significant four binary a general-purpose word register.	bit to
					9,0,0,0	designated by the Rs field. The conte of the destination are unaltered.	ents
					O W XA O		
					(x8) 0 + x8 (9h0		
					O O O		

S P/V DA H *

Z: Set to 1 if selected bit of destination operand is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

BIT test in a word (static)

BIT dst, B

BIT

Mode	Version	Mnemonic and Form BIT Rd, B			Clocks
R	NS, S	1,0,1,0,0,1,1,1	Rd	b	4
		BIT Rd↑, B			
IR	NS	0,0,1,0,0,1,1,1	Rd ≠ 0	b	8
		BIT RRd↑, B			_
IR	S	0,0,1,0,0,1,1,1	RRd ≠ 0	b	8
		BIT LABEL, B			,
DA	NS	0,1,1,0,0,1,1,1	0,0,0,0 RESS	b	10
			1200		
		BIT LABSSO, B	0.0.0.0	b	1
DA	SSO	0 SEGMENT	OFFSET		11
		BIT LABEL, B			
_ IId-est		0,1,1,0,0,1,1,1	0,0,0,0	b	
	SLO SLO		SET		13
		Miles a statement of the viscous of	SEI		_
		BIT LABEL (Rx), B	Rx ≠ 0	b	7
X	NS	ADDI	RESS	U	11
		BIT LABSSO (Rx), B			
X	000	0,1,1,0,0,1,1,1	Rx ≠ 0	b	11
^	SSO	0 SEGMENT	OFFS	ET] ''
		BIT LABEL (Rx), B		1	
V		0 1 1 1 1 0 1 0 1 1 1 1 1	Rx ≠ 0	b	8
X	SLO	1 SEGMENT			3 14

Description

A bit in the word destination is tested, and the Z flag is affected as shown below. The destination is determined by the applicable addressing mode, while the bit to be tested is specified by the binary value of the b field of the instruction. The contents of the destination are not altered.

Operation

Z flag←word dst<b bit>

Assembler Notation

The assembler notation B is a numeric expression which is assembled into a binary value in the b field of the instruction. The range of b is zero through 15, and b = B. Specifying a B outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

Z S P/V DA H Z: Set to 1 if specified bit of destination word is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

BITB

BIT test in a byte (dynamic)

BIT Rbd, Rs

BITB

BIT Rbd, Rs of S O O O O O O O O O O O O O O O O O O	8 8	8	0 + 6A 10 0 + 6A 10 0 0 0 0 0 0 0 0	Z flag←Rbd bit specif	ied in Rs(0:2)	
			0 % bAR [
			10.0.0.0 0AESE			
					GH1	AC
				Description		
			9,0,0,0)1 130 5,0,0,0)1 T281		e Z flag is n byte operan gister designat nstruction. The ed from a bina nificant three b	d is ed bit ary its of
			Print Print O	The contents of the des unaltered.).):
			O + XR I			
			1 Px = 0			
				The st sembler Notation ORESS The st sembler notation The st sembler notation ORESET The st sembler notation Swarpersion which is as To semble in the billion of the range of the standard or the sembler and the standard or the standar	a general-purpose word nated by the Rs field of The contents of the desunaltered. Indicator reidments as soft as all derivers as some services of the desunaltered. Indicator reidments as soft as all derivers as some services of the desunaltered. Indicator reidments as soft as all derivers as some services of the desunaltered. Indicator reidments as soft as all derivers as some services of the desunaltered. Indicator reidments as soft as all derivers as some services as som	a general-purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered. But a general purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered. But a general purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered. But a general purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered. But a general purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered. But a general purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered. But a general purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered. But a general purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered. But a general purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered. But a general purpose word register designated by the Rs field by the R

C Z S P/V DA H

Z: Set to 1 if selected bit of destination operand is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

D	ITD	dst.	
	I I D	ust.	·E

Mode	Version	Mnemonic and Form BITB Rbd, B	Clocks	Operation Z flag←byte dst <b bit="">		
7 198	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 Rbd b	4			
		PO SEGNENTdat<0				
R	NS	BITB Rd↑, B $ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	8	CALL Patr		
R	S	BITB RRd↑, B 0 0 1 0 0 1 1 1 0 RRd ≠ 0 b	8	LITERALIS SM STATE		
		BITB LABEL, B		Lateration of State o		
DA	NS	0 1 1 0 0 1 1 0 0 0 0 b ADDRESS	10	Description A bit in the byte destination is tested, and		
DA	SSO	BITB LABSSO, B 0 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0	0,11,0	the Z flag is affected as shown below. The destination is determined by the applicable addressing mode. The bit to be tested is determined by the binary value of the least significant three bits.		
DA	SLO	BITB LABEL, B 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0	13	value of the least significant three bits of the b field of the instruction. The content of the destination are not altered.		
	SLO	BITB LABEL (Rx), B 0 1 1 0 0 1 1 0 Rx ≠ 0 b ADDRESS	0 h 11 d 0	Assembler Notation The assembler notation B is a numeric expression which is assembled into a		
e desig- unation	NS	BITB LABSSO (Rx), B 0 1 1 1 0 0 1 1 1 0 Rx ≠ 0 b 0 SEGMENT OFFSET	11,	binary value, b, in the instruction. The range of b is zero through 7, and b = B. Specifying a B outside of the allowable range produces an assembler error.		
×	SSO	BITB LABEL (Rx), B 0 1 1 0 0 1 1 0 Rx ≠ 0 b 1 SEGMENT OFFSET	0 - 14	CALL LABEL (AX) LO 1 D 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		
				0		

Flags



Z: Set to 1 if specified bit of destination byte is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CALL

CALL subroutine

CALL dst

CALL

Mode	Version	Mnemonic and Form d > rab elyd = pad S	bdPl (d	Operation (segmented) R15 < 0:15>←R15<0:15> −2 (RR14<0:22>)←Updated PC OFFSET R15<0:15>←R15<0:15> −2 (RR14)<0:22>)←PC SEGMENT PC SEGMENT←dst<24:30> PC OFFSET←dst<0:15>
ID.	1 1 1 917	CALL Rd↑	1 4 6 F	Operation (non-segmented)
IR	NS	0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 1 Rd 0 ₁ 0 ₁ 0 ₁ 0	10	R15<0:15>←R15<0:15>-2 (R15<0:15>)←Updated PC
IR	S	CALL RRd↑ 0,0,0,1,1,1,1,1,1 RRd 0,0,0,0	15	PC←dst<0:15> ☐ In the system mode the system stack
		CALL LABEL		pointer (R15' or RR14') is used instead of the normal stack pointer.
DA	NS	0,1,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0 ADDRESS	12	
		CALL LABSSO 0,1,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0		AC SSO CITITO DISTRICT
DA	SSO	0 SEGMENT OFFSET	18	Description The program return address (i.e., the
DA	SLO	CALL LABEL 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0	20	updated contents of PC) is pushed onto the stack addressed by the implied stack pointer register (R15 non-segmented, RR14 segmented). The new program counter address is then loaded to transfer
X of	NS	CALL LABEL (Rx) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	13 ARC	control to the subroutine. The new address is determined by the applicable addressing mode.
		CALL LABSSO (Rx)		In the IR mode R0 (or RR0) can be designated as the general-purpose destination
X	SSO	0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	18	register.
		CALL LABEL (Rx)		8 (xA) J38AL 8TIB
X	SLO	0,1,0,1,1,1,1,1 Rx ≠ 0 0,0,0,0 1 SEGMENT OFFSET	21	X SSO 1 SEGMENT

C Z S P/V DA H Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

- = Unaffected1 = Set0 = Cleared

* = Conditional - see description

CALL subroutine relative

CALR

CALR LAB

Mode	Version	Mnemonic and For CALR LAB	m zalacia		Clocks	Operation (segmented) R15 <0:15>←R15<0:15>-2
RA	NS, S	1,1,0,1	DISPLACE	MENT	10, 15	(RR14<0:22>)←Updated PC OFFSET R15<0:15>←R15<0:15>-2 (RR14<0:22>)←PC SEGMENT PC OFFSET←Updated PC OFFSET- 2x displacement
						Operation (non-segmented) R15<0:15>←R15<0:15>−2 (R15<0:15>)←Updated PC
					L bas I	PC←Updated PC−2x displacement In the system mode the system stack pointer (R15' or RR14') is used instead of the payment stock pointer.
						the normal stack pointer.
						Description The program return address is pushed onto the stack addressed by the implied stack pointer register (R15 non-
						segmented, RR14 segmented in the normal mode). The signed 12-bit displacement field of the instruction is sign extended and left shifted (word aligned) before being subtracted from the updated
						PC (return address). The result is then loaded into the program counter to produce a jump address. The program counter segment number remains
						unaltered. The range of the relative call is -2047 to +2048 words with respect to the updated PC.
						Assembler Notation The label LAB is an address which is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside of the allowable range produces an assembler error.
Flags C Z	S P/\	/ DA H		are not affect		Flags C Z S PIV DA H

5-23

CLR

CLEAR word

CLR dst

CLR

Mode	Version	Mnemonic and Form CLR Rd	Clocks	Operation dst<0:15> ← 0
R	NS, S	1,0,0,0,1,1,0,1 Rd 1,0,0,0	7	
IR	NS	CLR Rd↑ 0_10_10_10_11_11_0_11 Rd 1_10_10_10	8	
IR	S	CLR RRd↑ 0_10_10_10_11_1_1_0_1 RRd 1_1_0_1_0	8	
		CLR LABEL		Description
DA	NS	0,1,0,0,1,1,0,1,0,0,0,0,1,0,0,0 ADDRESS	11	The 16 bits of the specified destination word are replaced with zeros. The original contents of the destination are
DA	SSO	CLR LABSSO 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 0 SEGMENT OFFSET	12	lost. The destination is determined by the applicable addressing mode. In the IR mode R0 (or RR0) can be designated the designation of the destination of the destination of the destination are destination.
DA	SLO	CLR LABEL 0 1 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 0 0 0 1 0	14	nated as the general-purpose destination register.
X	NS	CLR LABEL (Rx) $0_11_10_10_11_10_11_1 \text{ Rx} \neq 0 1_10_10_10_10_10_10_10_10_10_10_10_10_10$	12	
X	SSO	CLR LABSSO (Rx) 0 1 1 0 1 1 1 1 0 1 1 Rx ≠ 0 1 1 0 1 0 0 0 SEGMENT OFFSET	12	
X	SLO	CLR LABEL (Rx) 0 1 1 0 1 0 1 1 1 1 0 1 Rx ≠ 0 1 1 0 1 0 1 0 1 0 1 1 SEGMENT OFFSET	15	
		Some Bend recently disease of the second sec		

Flags					
C	Z	S	P/V	DA	Н
					1

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared * = Conditional - see description

CLEAR byte

CLRB dst

CLRB

Mode	Version	Mnemonic and Form	Clocks	Operation		
		CLRB Rbd		dst<0:7> ← 0		
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 Rbd 1 ₁ 0 ₁ 0 ₁ 0	7	10,1,1,0,0,0,0,1]		
		CLRB Rd↑		turi Bido		
IR	NS	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 Rd 1 ₁ 0 ₁ 0 ₁ 0	8	0.1.1.0.0.0.0.0		
		CLRB RRd↑		ISBR MOO		
IR	S	0,0,0,0,1,1,0,0 RRd 1,0,0,0	8	.0.1.1.0.0.0.0.01		R
		CLRB LABEL		Description		
DA	NS	0,1,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0 ADDRESS	11	The eight bits of the byte are replaced with		
		CLRB LABSSO		contents of the destination is determined	nation are lost.	
DA	SSO	0,1,0,0,1,1,0,0,0,0,0,0,1,0,0,0	12	applicable addressin		
-plast r	15 PERO (CER DE	0 SEGMENT OFFSET	0 12	In the IR mode R0 (d		
		CLRB LABEL		nated as the general register.	-purpose desti	nation
DA	SLO	0,1,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0 1 SEGMENT	14	register.		
	OLO	OFFSET	130	6		
		CLRB LABEL (Rx)		COM LABEL (Rx)		
X	NS	0,1,0,0,1,1,0,0 Rx ≠ 0 1,0,0,0 ADDRESS	12	0.1.1.0.0.1.0		
		CLRB LABSSO (Rx)		NA CESSAL MOS		
		CLID LABSSO (IX)		120 F R 1 TO THE PROPERTY OF THE PARK I		
Y	022	$0_11_10_10_11_11_10_10$ Rx $\neq 0$ $1_10_10_10$	10	0.1,0,0,1,1,0,		
X	SSO		12	O SEGMENT		
×	SSO	0 1 1 0 1 0 1 1 1 1 0 1 0 Rx ≠ 0 1 1 0 1 0 1 0 0 0 SEGMENT OFFSET CLRB LABEL (Rx)	12	COM LABEL (88)		
X	SSO	0 1 1 0 1 0 1 1 1 1 0 0 Rx ≠ 0 1 1 0 1 0 0 0 SEGMENT OFFSET	12	THEMESE 10		



C Z S P/V DA H Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

COM

COMPLEMENT word

COM

COM dst

Mode	Version	Mnemonic and Form COM Rd	Clocks	Operation dst<0:15> ← dst<0:1	5>.	
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 Rd 0 ₁ 0 ₁ 0 ₁ 0	bdF 7] 0			
IR.	NS	COM Rd1 01010101111011 Rd 0101010	12			
., .	140		0.1-			
IR	S	COM RRd↑ 0,0,0,0,1,1,1,0,1 RRd 0,0,0,0	12	CURB RROL		SH
		COM LABEL		Description		
DA	NS	0 1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 ADDRESS	15	The contents of the de	ented. The or	iginal
		COM LABSSO		contents of the destination operand is		
DA	SSO	0 1 1 0 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 0	16	applicable addressing		90
		(c) 931 e0011 11 e111 11	0	In the IR mode R0 (or nated as the general-p		
		COM LABEL 0,1,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0	0.0.010	register.	urpose destir	lation
DA	SLO	1 SEGMENT OFFSET	18			
			1361			
		COM LABEL (Rx) $ \begin{bmatrix} 0_1 & 1_1 & 0_1 & 0_1 & 1_1 & 1_1 & 0_1 & 1_1 & 0_1 $				
X	NS	ADDRESS	16	GA		
		COM LABSSO (Rx)				
X	SSO	$0_1 1_1 0_1 0_1 1_1 1_1 0_1 1$ $Rx \neq 0$ $0_1 0_1 0_1 0$	16			
		0 SEGMENT OFFSET	0	[0] SIGMENT		
		COM LABEL (Rx) $ \begin{bmatrix} 0_1 & 1_1 & 0_1 & 0_1 & 1_1 & 1_1 & 0_1 & 1_1 & 0_1 $	n is an In			
X	SLO	1 SEGMENT	19			
		OFFSET	71 - 72 - 73			



С	Z	S	P/V	DA	Н
_	2012	*	_	_	-1

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

^{– =} Unaffected

^{1 =} Set

^{0 =} Cleared

^{* =} Conditional - see description

COMPLEMENT byte

COMB

COMB dst

	Wilemonic and Form	Clocks	Operation dst<0:7> ← dst<0:7>
NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 Rbd 0 ₁ 0 ₁ 0 ₁ 0	8.7.0	1011,110,010,11 8,89
NS	COMB Rd↑ 01010101111010 Rd 0101010	12	
6	COMB RRd↑	10	
3		12	Description
NS	011101011110100101010101010 ADDRESS	15	The contents of the destination byte operand are complemented. The original contents of the destination are lost. The
SSO	COMB LABSSO 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0	16	destination operand is determined by the applicable addressing mode.
SLO	COMB LABEL 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0		nated as the general-purpose destination register.
	Complement P/V Flag PY or GV		
NS	0_11_0_0_1_1_1_0_0 Rx ≠ 0 0_0_0_0 ADDRESS	16	
	COMB LABSSO (Rx)		ssembler Note Ion
SSO	0 1 1 0 1 0 1 1 1 1 0 0		ne assembler notation LIST refers to a list of the Same fill. OV. Note that Travery and OV affect the Same fill.
SLO	COMB LABEL (Rx) 0 1 1 0 1 0 1 1 1 1 0 1 0 Rx ≠ 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	19	
	NS S SSO SLO NS SSO	COMB Rbd 110101111100 Rbd 0101010 NS 010101111100 Rd 0101010 COMB RRd↑ 01010111100 RRd 010100 COMB LABEL 01110111100 RRd 0101000 NS 011101111000 01010000 0101000 ADDRESS COMB LABSSO SSO 01110111000000000000 0FFSET COMB LABEL 0110011110000000000000000000000000000	COMB Rbd NS, S COMB Rbd Table Table

Flags

С	Z	S	P/V	DA	Н
-	*	*	*	-	-

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 if parity of result is even. Reset otherwise.

0 = Cleared

5

^{- =} Unaffected

^{1 =} Set

^{* =} Conditional - see description

COMFLG

COMPLEMENT FLAGS

COMFLG

COMFLG LIST

Mode	Version	Mnemonic and Form COMFLG LIST	Clocks	Clocks	Operation FCW <c;z;s;p v="">←FCW<c;z;s;p th="" v<=""><th>sheld /></th></c;z;s;p></c;z;s;p>	sheld />
-	NS, S	1,0,0,0,1,1,0,1,	C,Z,SPV0,1,0	1 67	(See description below)	
					COMB Flor	
					NS [0,0,0,0,1,1,0,0	
					TORR SMOO	
					101110000	

Description

The CPU flags C, Z, S and P/V are complemented or unaltered, according to the bit settings in the instruction field as described in the table below.

Instruction Bit	If = 0	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,	Assembler Notation
uplateneg 7 dt es bols	No Effect	Complement C Flag	CY ALI BMC
6	No Effect	Complement Z Flag	ZR D
5	No Effect	Complement S Flag	SGN
4	No Effect	Complement P/V Flag	PY or OV

Assembler Notation

The assembler notation LIST refers to a list of any or all of the following reserved words, separated by commas: CY, ZR, SGN, PY or OV. Note that PY and OV affect the same flag (P/V).

Flags



See above.

H: Undefined.

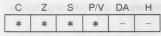
- = Unaffected

1 = Set

0 = Cleared

Mode	Version	Mnemonic and Form CP Rd, Rs	Clocks	Operation Use the result of Rd<0		15>
3	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 Rs Rd	4	to set flags.	244	10/
		CP Rd. IM		1990		
М	NS, S	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd	7	OP RR41, IM		
		OPERAND DE COMPANDO DE COMPAND		1,0,1,1,0,0,0,0,0		
R	NS	CP Rd, Rs↑ 0,0,0,0,1,0,1,1 Rs ≠ 0 Rd	0,070	OP LABEL IM		
П	NS			IGGA		
R	S	CP Rd, RRs↑ 0 0 0 0 1 0 1 1 RRs ≠ 0 Rd	OMAF 7	1990		
	3			Description		
		CP Rd, LABEL 0 1 0 0 1 0 1 1 0 0	0,0,0,0	The source word opera	and is compar	he
DA	NS	ADDRESS	9	by subtraction with the	contents of a	
		CP Rd, LABSSO		general-purpose word by the Rd field of the i		
DA	SSO	0 1 1 0 1 0 1 1 0 1 1 1 0 1 0 1 0 1 Rd	0 0 10	source operand is dete	ermined by the	9
(10)	tone dissiblined and the univ	0 SEGMENT OFFSET		applicable addressing source contents and d		
		CP Rd, LABEL		are unaltered.	estination con	torita
DA	SLO	0,1,0,0,1,0,1,1 0,0,0,0 Rd 1 SEGMENT	12	CP LABEL (Rx), IM		
		OFFSET		10,1,1,0,0,1,0		
		CP Rd, LABEL (Rx)		BOA		
<	NS	0,1,0,0,1,0,1,1 Rx ≠ 0 Rd	10			
<	NS	ADDRESS		OP LABSSO (Rx), IM		
×		ADDRESS CP Rd, LABSSO (Rx)		0,1;0;0;1;1;0;1 0 SEGMENT		
×	NS SSO	ADDRESS		0,1;0;0;1;1;0;1 0 SEGMENT		
		ADDRESS CP Rd, LABSSO (Rx) 0 1 0 0 1 0 1 1 Rx ≠ 0 Rd	0 x vA HO 10 /AF	OF LABEL (Rx), IM		
<	SSO	ADDRESS CP Rd, LABSSO (Rx) $0 \mid 1 \mid 0 \mid 0 \mid 1 \mid 0 \mid 1 \mid 1$ $Rx \neq 0$ Rd $0 \mid SEGMENT \mid OFFSET$ CP Rd, LABEL (Rx) $0 \mid 1 \mid 0 \mid 0 \mid 1 \mid 0 \mid 1 \mid 1$ $Rx \neq 0$ Rd	0 × xA 10 //AF	OF LABEL (Rx) IM OP LABEL (Rx) IM OF LABEL (Rx) IM		
		ADDRESS CP Rd, LABSSO (Rx) 0 1 1 0 1 1 0 1 1 1	0 x vA HO 10 /AF	OF LABEL (Rx), IM		

Flags



- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

- C Z S P/V DA H C See to n carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
 - Z: Set to 1 if result is zero. Reset otherwise.
 - S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 on arithmetic overflow. Reset otherwise.

CP

COMPARE IMMEDIATE word with memory

CP

CP dst, IM

	Version	Mnemonic and Form CP Rd↑, IM	Clocks	Operation Use result of dst<0:15>-src<0:15>
IR	NS	0 0 0 0 1 1 0 1 Rd 0 0 0 1 1 OPERAND	11	to set flags (see below).
		CD DDda IM		CP Rd, IM
IR	S	CP RRd↑, IM 0 0 0 0 1 1 0 1 RRd 0 0 0 1	14 MAR	M WS. S 0,0,0,1,1,0,1,
		CP LABEL, IM		CP Rd Rs
DA	NS	0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ADDRESS	0 + 48	NS (0,0,0,0,1,1
DA	140	OPERAND	14	falls.at
		CD LARSCO IM		-1,5,1,0,0,0,0,0 B
		CP LABSSO, IM 0		Description Page
DA	SSO	0 SEGMENT OFFSET	15	The immediate source word operand is
		0. 2.0		compared with the destination word
		CP LABEL, IM		operand. The comparison is achieved by subtraction. The destination operand is
				determined by the applicable addressing
		1 SEGMENT		mode. The contents of the destination
DA	SLO	I HE ZITOTADO BOTUCE OFFSET	17	operand are unaltered and the only
		OPERAND		action is to set the flags as described below.
		CP LABEL (Rx), IM		Transpage 1 1 A F
		$0_1 1_1 0_1 0_1 1_1 1_1 0_1 1$ Rx $\neq 0$ $0_1 0_1 0_1 1$		In the IR mode R0 (or RR0) can be designated as the general-purpose destination
X	NS	ADDRESS	15	register.
		OPERAND		F. F. O. L. O. O. E. O.
		CP LABSSO (Rx), IM		IGA X
		$0_1 1_1 0_1 0_1 1_1 1_1 0_1 1$ $Rx \neq 0$ $0_1 0_1 0_1 1$		(A) (2000 - 1 A) (200
X	SSO	0 SEGMENT OFFSET	15	CPR4 LASSO (Ro)
		OPERAND		X SSO 0 SEGMENT
		CP LABEL (Rx), IM		
		$0_1 1_1 0_1 0_1 1_1 1_1 0_1 1$ $Rx \neq 0$ $0_1 0_1 0_1 1_1$		OP BULL ABEL (Rx)
Χ	01.0	1 SEGMENT	18	X SLO 11 SEGMENT
\wedge	SLO	OFFSET	10	80
		OPERAND		

C		S	P/V	DA	H
*	*	*	*	-	-

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Reset on carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
 - Z: Set to 1 if result is zero. Reset otherwise.
- S: Set to 1 if result is negative. Reset otherwise.
- P/V: Set to 1 on arithmetic overflow. Reset otherwise.

CPB Rbd, src

Mode	Version	Mnemonic and Form	Clocks	Operation		
		CPB Rbd, Rbs		Use result of Rbd<0:	7>-src<0:7>	>
7	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 0 Rbs Rbd	4	to set flags.		
		CPB Rbd, IMb	340 313	7. OPERAND		
М	NS. S	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd] -	dNI ,15RA ERO		
IVI	NS, S	7 OPERAND 0 7 OPERAND 0	681 ⁷	0,0,0,0,1,1,0,		
		CPB Rbd, Rs↑		OP3 LABEL, IMb		
R	NS	$0_10_10_10_11_10_11_10$ Rs $\neq 0$ Rbd	7 0 6	10,1,1,0,0,1,0		
		CPB Rbd, RRs↑		7 OPERAND		
R	S	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 0 RRs ≠ 0 Rbd	7	GYIAM3 TO 1		
		CPB Rbd, LABEL	0.0.0.010	Description		
DA B	NS	0,1,0,0,1,0,1,0,0,0,0,0 Rbd	9	The source byte opera	and is compar	ed by
JA	NS no.	ADDRESS	190 4 110	subtraction with the co	ontents of a	
		CPB Rbd, LABSSO		general-purpose byte by the Rbd field of the		
DA -	200		10 00			
DA P	SSO		10	source operand is det applicable addressing	ermined by th mode. Both t	e he
DA	SSO	0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0	10	source operand is det applicable addressing source contents and o	ermined by th mode. Both t	e he
desig- ination	here unallered.	0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 Rbd	10	source operand is det applicable addressing	ermined by th mode. Both t	e he
desig- ination	here unallered.	0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 Rbd 0 SEGMENT OFFSET CPB Rbd, LABEL 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 Rbd 1 SEGMENT	10	source operand is det applicable addressing source contents and o	ermined by th mode. Both t	e he
	re unangred APIO) can ba purpose dest	0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 Rbd 0 SEGMENT OFFSET CPB Rbd, LABEL 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 Rbd	F3E1	source operand is det applicable addressing source contents and o are unaltered.	ermined by th mode. Both t destination con	e he
desig- ination	re unangred APIO) can ba purpose dest	0 1 1 0 1 1 0 1 1 0 0 1 0 1 0 0 0 0 0 0	12	source operand is det applicable addressing source contents and o are unaltered.	ermined by th mode. Both t	e he
DA	re unangred APIO) can ba purpose dest	0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0	12	source operand is det applicable addressing source contents and d are unaltered.	ermined by th mode. Both t destination con	e he
DA	SLO	0 1 1 0 1 1 0 1 1 0 0 1 0 1 0 0 0 0 0 0	12	source operand is det applicable addressing source contents and o are unaltered.	ermined by th mode. Both t destination con	e he
desig- ination	SLO	0 1 1 0 1 1 0 1 1 0 0 1 0 1 0 0 Rbd 0 SEGMENT OFFSET CPB Rbd, LABEL 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 Rbd 1 SEGMENT OFFSET CPB Rbd, LABEL (Rx) 0 1 1 0 1 1 0 1 1 0 Rx ≠ 0 Rbd ADDRESS CPB Rbd, LABSSO (Rx)	12	source operand is det applicable addressing source contents and d are unaltered.	ermined by th mode. Both t destination con	e he
DA (SLO	0 1 1 0 1 1 0 1 1 0 0 1 0 1 0 0 1 0	12	source operand is det applicable addressing source contents and d are unaltered.	ermined by th mode. Both t destination con	e he
DA X	SLO	0 1 1 0 1 1 0 1 1 0 0 1 0 1 0 0 Rbd 0 SEGMENT OFFSET CPB Rbd, LABEL 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 Rbd 1 SEGMENT OFFSET CPB Rbd, LABEL (Rx) 0 1 1 0 1 1 0 1 1 0 Rx ≠ 0 Rbd ADDRESS CPB Rbd, LABSSO (Rx)	12	source operand is det applicable addressing source contents and o are unaltered.	ermined by th mode. Both t destination con	e he
DA K	SLO	0 1 1 0 1 1 0 1 1 0 0 1 0 1 0 0 1 0	12	source operand is det applicable addressing source contents and of are unaltered.	ermined by th mode. Both t destination con	e he
DA	SLO	0,1,0,0,1,0,1,0 0,0,0,0 Rbd 0 SEGMENT OFFSET CPB Rbd, LABEL 0,1,0,0,1,0,1,0,0,0,0 Rbd 1 SEGMENT OFFSET CPB Rbd, LABEL (Rx) 0,1,0,0,1,0,1,0,1,0 Rx ≠ 0 Rbd ADDRESS CPB Rbd, LABSSO (Rx) 0,1,0,0,1,0,1,0,1,0 Rx ≠ 0 Rbd 0 SEGMENT OFFSET	12	source operand is det applicable addressing source contents and o are unaltered.	ermined by th mode. Both t destination con	e he



- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- H C: Reset on carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
 - Z: Set to 1 if result is zero. Reset otherwise.
 - S: Set to 1 if result is negative. Reset otherwise.
- P/V: Set to 1 on arithmetic overflow. Reset otherwise.

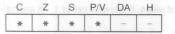
CPB

COMPARE IMMEDIATE byte with memory

CPB

CPB dst, IMb

/lode	Version	Mnemonic and Form CPB Rd↑, IMb	Clocks	Operation Use result of dst<0:7>-src<0:7>
R	NS	0 0 0 0 1 1 1 1 0 0 Rd 0 0 0 1 1 7 OPERAND 0	11	to set flags.
		CPB RRd↑, IMb		CM1 bd9 890
R	S	0101011111010 RRd 0101011	9011	M NE. 6 7 OP BRAND
		CPB LABEL, IMb		OPB Rbd. Ref.
DA	NS	0,1,0,0,1,1,0,0,0,0,0,0,0,0,1 ADDRESS	14	rijo, trejojojo
		7 OPERAND 0 7 OPERAND 0		1aRS, bdR R90
		CPB LABSSO, IMb 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0		Description
AC	SSO	0 SEGMENT OFFSET	15	The immediate source byte operand is
		7 OPERAND 0 7 OPERAND 0		compared by subtraction with the destination byte operand. The destination
		CPB LABEL, IMb		operand is determined by the applicable
		0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,1		addressing mode. The contents of the destination operand are unaltered.
DA	SLO	1 SEGMENT OFFSET	17	In the IR mode R0 (or RR0) can be desig-
		7 OPERAND 0 7 OPERAND 0		nated as the general-purpose destination
		CPB LABEL (Rx), IMb		register.
,		$0_11_10_10_11_11_10_10$ Rx $\neq 0$ $0_10_10_11$		
X	NS	ADDRESS 7 OPERAND 0 7 OPERAND 0	15	CPS Rbu, LABEL (Rx
		A L 109 LL		11.0,1.0,0,1.0
		CPB LABSSO (Rx), IMb 0 1 1 0 1 0 1 1 1 1 0 0		GA
X	SSO	0 SEGMENT OFFSET	15	CRB Mb LABSSD
		7 OPERAND 0 7 OPERAND 0		X SSD 0 11011
		CPB LABEL (Rx), IMb		
		$0_1 1_1 0_1 0_1 1_1 1_1 0_1 0 Rx \neq 0 0_1 0_1 0_1 1$		(d) 23 ded 1 dR dRO
X	SLO	1 SEGMENT OFFSET	18	TVENDE TT OLD X
		7 OPERAND 0 7 OPERAND 0		0



- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Reset on carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
- Z: Set to 1 if result is zero. Reset otherwise.
- S: Set to 1 if result is negative. Reset otherwise.
- P/V: Set to 1 on arithmetic overflow. Reset otherwise.

CPD

COMPARE register to memory word, autodecrement

CPD

CPD Rd, src, Rc, CC

Mode	Version	Mnemonic and Form			Clocks	Operation
		CPD Rd, Rs1, Rc, CC				If result of Rd<0:15>-src<0:15> meets
IR	NO	1,0,1,1,1,0,1,1	Rs	1,0,0,0	20	CC condition in instruction, then Z flag←
In	NS	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd	CC	20	Rs<0:15>←Rs<0:15>-2
		CPD Rd, RRs1, Rc, CC				Rc<0:15>←Rc<0:15>-1
IR	S	1,0,1,1,1,0,1,1	RRs	1,0,0,0	20	F. O. F. F. D. O.
In	5	0,0,0,0 Rc	Rd	CC	20	8 10 0 0 0 1

Description

The source word operand is compared to the destination word operand by subtraction. The destination operand is the contents of the general-purpose word register designated by the Rd field of the instruction. The source operand is a word in memory addressed by the generalpurpose register designated by the Rs (or RRs) field of the instruction. Both source and destination operands are unaltered, and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are decremented by two.

R0 can be designated as the generalpurpose source register.

Flags

DA *

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise. P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPDB

COMPARE register to memory byte, autodecrement

CPDB

CPDB Rbd, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks	Operation
		CPDB Rbd, Rs1, Rc, CC		If result of Rbd<0:7>-src<0:7> meets
IR	A non inches	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 Rs 1 ₁ 0 ₁ 0 ₁ 0	00	CC condition in instruction, then Z flag←1.
In	NS	0101010 Rc Rbd CC	20	Rs<0:15>←Rs<0:15>-1
		CPDB Rbd, RRs↑, Rc, CC		Rc<0:15>←Rc<0:15>−1
ID		1,0,1,1,1,0,1,0 RRs 1,0,0,0	III RAS	0.0.11.10.1
IR	S	0,0,0,0 Rc Rbd CC	20	0,0,0,0 Rc

Description

The source byte operand is compared to the destination byte operand by subtraction. The destination operand is the contents of the general-purpose byte register designated by the Rbd field of the instruction. The source operand is a byte in memory addressed by the general-purpose register designated by the Rs (or RRs) field of the instruction. Both source and destination operands are unaltered, and the only action is to set the flags. The contents of the generalpurpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are decremented by one.

R0 can be designated as the general-purpose source register.

Flags

C Z S P/V DA H

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPDR

COMPARE register to memory word, autodecrement and repeat

CPDR

CPDR Rd, src, Rc, CC

Mode	Version	Mnemonic and Form			Clocks
		CPDR Rd, Rs1, Rc, CC			
IR	ALC: HE	1,0,1,1,1,0,1,1	Rs	1,1,0,0	44 0-4
In	NS	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd	CC	11 + 9n*
		CPDR Rd, RRs1, Rc, CC			
IR	6	1,0,1,1,1,0,1,1	RRs	1,1,0,0	11 + 9n*
In	5	0,0,0,0 Rc	Rd	CC	11 + 9h*

^{*}n is the number of iterations.

Operation and an address of the state of the

If Rd<0:15>-src<0:15> meets CC condition in instruction, then Z flag←1. Rs<0:15>←Rs<0:15>-2 Rc<0:15>←Rc<0:15>-1 Repeat until termination.

Description

The source word operand is compared to the destination word operand by subtraction. The source operand is a word in memory addressed by the generalpurpose register designated by the Rs (or RRs) field of the instruction. The destination operand is the contents of the general-purpose word register designated by the Rd field of the instruction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are decremented by two, and the operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source register.

Flags

C Z S P/V DA H

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise. P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

Unaffected

1 = Set

0 = Cleared

CPDRB

COMPARE register to memory byte, autodecrement and repeat

CPDRB

CPDRB Rbd, src, Rc, CC

Mode	Version				Clocks	Operation 600M
		CPDRB Rbd, Rs1, Rc, CC				If Rbd<0:7>-src<0:7> meets CC
IR	NS	1,0,1,1,1,0,1,0	Rs	1,1,0,0	11 + 9n*	condition in instruction, then Z flag←1.
11.1	INO	0,0,0,0 Rc	Rbd	CC	11 7 311	Rs<0:15>←Rs<0:15>-1
		CPDRB Rbd, RRs1, Rc, C	С		20	Rc<0:15>←Rc<0:15>-1 Repeat until termination.
IR	0	1,0,1,1,1,0,1,0	RRs	1,1,0,0	44 . 0-4	
III	S	0,0,0,0 Rc	Rbd	CC	11 + 9n*	

^{*}n is the number of iterations.

Description

The source byte operand is compared to the destination byte operand by subtraction. The source operand is a byte in memory addressed by the generalpurpose register designated by the Rs (or RRs) field of the instruction. The destination operand is the contents of the general-purpose byte register designated by the Rbd field of the instruction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the generalpurpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are decremented by one, and the operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source register.

Flags

C Z S P/V DA H

H Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

CPI

COMPARE register to memory word, autoincrement

CPI

CPI Rd, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks	Operation If result of Rd<0:15>-src<0:15> meets
IR	NS	110111110111 Rs 0101010 0101010 Rc Rd CC	20	CC condition in instruction, then Z flag←1. Rs<0:15>←Rs<0:15>+2
IR	S	CPI Rd, RRs1, Rc, CC 110111110111 RRs 0101010 0101010 Rc Rd CC	20	Rc<0:15>←Rc<0:15>-1

Description

The source word operand is compared to the destination word operand by subtraction. The destination operand is the contents of the general-purpose word register designated by the Rd field of the instruction. The source operand is a word in memory addressed by the generalpurpose register designated by the Rs (or RRs) field of the instruction. Both the source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are incremented by two.

R0 can be designated as the generalpurpose source register.

Flags

C Z S P/V DA H

H Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

CPIB

COMPARE register to memory byte, autoincrement

CPIB

CPIB Rbd, src, Rc, CC

Mode	Version	Mnemonic and Form CPIB Rbd, Rs↑, Rc, CC			Clocks
IR	NS	1,0,1,1,1,0,1,0 0,0,0,0 Rc	Rs Rbd	0 ₁ 0 ₁ 0 ₁ 0 CC	20
IR	S	CPIB Rbd, RRs1, Rc, CC	RRs Rbd	0 ₁ 0 ₁ 0 ₁ 0	20

Operation

If result of Rbd<0:7>-src<0:7> meets CC condition in instruction, then Z flag \leftarrow 1. Rs<0:15> \leftarrow Rs<0:15>+1 Rc<0:15> \leftarrow Rc<0:15>-1

Description

The source byte operand is compared to the destination byte operand by subtraction. The destination operand is the contents of the general-purpose byte register designated by the Rbd field of the instruction. The source operand is a byte in memory addressed by the generalpurpose register designated by the Rs (or RRs) field of the instruction. Both the source and destination operands are unaltered, and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are incremented by one.

R0 can be designated as the general-purpose source register.

Flags

C Z S P/V DA H

DA H Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPIR

COMPARE register to memory word, autoincrement and repeat

CPIR

CPIR Rd, src, Rc, CC

Mode	Version	Mnemonic and Form			Clocks
		CPIR Rd, Rs1, Rc, CC	اساليك		
IR	NS	1,0,1,1,1,0,1,1	Rs	0,1,0,0	11 + 9n
In	NS	0,0,0,0 Rc	Rd	CC	11 + 9h
		CPIR Rd, RRs1, Rc, CC			
IR	0	1,0,1,1,1,0,1,1	RRs	0,1,0,0	44 . 0-
In	5	0,0,0,0 Rc	Rd	CC	11 + 9n

Operation

If Rd<0:15>-src<0:15> meets CC condition in instruction, then Z flag←1. Rs<0:15>←Rs<0:15>+2 Rc<0:15>←Rc<0:15>-1 Repeat until termination.

*n is the number of iterations.

Description

The source word operand is compared to the destination word operand by subtraction. The source operand is a word in memory addressed by the generalpurpose register designated by the Rs (or RRs) field of the instruction. The destination operand is the content of the general-purpose word register designated by the Rd field of the instruction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the generalpurpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are incremented by two. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source register.

Flags

C Z S P/V DA H

H Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

CPIRB

COMPARE register to memory byte, autoincrement and repeat

CPIRB

CPIRB Rbd, src, Rc, CC

Mode	Version	Mnemonic and Form CPIRB Rbd, Rs1, Rc, CC			Clocks
IR	NS	1,0,1,1,1,0,1,0 0,0,0,0 Rc	Rs Rbd	0 ₁ 1 ₁ 0 ₁ 0 CC	11 + 9n*
IR	S	CPIRB Rbd, RRs1, Rc, C	RRs Rbd	0 ₁ 1 ₁ 0 ₁ 0	11 + 9n*

^{*}n is the number of iterations.

Operation

If Rbd<0:7>-src<0:7> meets CC condition in instruction, then Z flag \leftarrow 1. Rs<0:15> \leftarrow Rs<0:15>+1 Rc<0:15> \leftarrow Rc<0:15>-1 Repeat until termination.

Description

The source byte operand is compared to the destination byte operand by subtraction. The source operand is a byte in memory addressed by the generalpurpose register designated by the Rs (or RRs) field of the instruction. The destination operand is the contents of the general-purpose byte register designated by the Rbd field of the instruction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the generalpurpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are incremented by one, and the operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source register.

Flags

C Z S P/V DA H

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPL

COMPARE register with long word

CPL

CPL RRd, src

Mode	Version	Mnemonic and Form	Clock
atasam -		CPL RRd, RRs	
R	NS, S	1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 RRs RRd	8
		CPL RRd, IMl	
		0,0,0,1,0,0,0,0,0,0,0,0 RRd	
IM	NS, S	31 OPERAND 16	14
		15 OPERAND 0	
		CPL RRd, Rs↑	
IR	NS	$0_10_10_11_10_10_10_10$ Rs $\neq 0$ RRd	14
		CPL RRd, RRs↑	
IR	S	0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 RRs ≠ 0 RRd	14
		CPL RRd, LABEL	
DA	NS	0,1,0,1,0,0,0,0,0,0,0,0,0 RRd ADDRESS	15
		ADDRESS	
		CPL RRd, LABSSO	
DA	SSO	0 ₁ 1 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 RRd	16
	000	0 SEGMENT OFFSET	
		CPL RRd, LABEL	
		0,1,0,1,0,0,0,0,0,0,0,0 RRd	
DA	SLO	1 SEGMENT	18
		OFFSET	
		CPL RRd, LABEL (Rx)	
X	NS	$0_11_10_11_10_10_10_10$ Rx $\neq 0$ RRd	16
^	NS	ADDRESS	10
		CPL RRd, LABSSO (Rx)	
.,		$0_1 1_1 0_1 1_1 0_1 0_1 0_1 0$ RRd	
X	SSO	0 SEGMENT OFFSET	16
		ODL DD4 LABEL (D.)	
		CPL RRd, LABEL (Rx) 0	
X	SLO	0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 RX ≠ 0 RRd 1 SEGMENT	19
,,	OLO	OFFSET	, 0

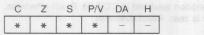
Description

The source long word operand is compared by subtraction with the contents of a general-purpose register pair designated by the RRd field of the instruction. The source operand is determined by the applicable addressing mode. Both the source contents and destination contents are unaltered.

Operation
Use result of RRd<0:31>-src<0:31>

to set flags.

Flags



- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Reset on carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
- Z: Set to 1 if result is zero. Reset otherwise.
- S: Set to 1 if result is negative. Reset otherwise.
- P/V: Set to 1 on arithmetic overflow. Reset otherwise.

5

CPSD

COMPARE word strings in memory, autodecrement

CPSD

CPSD dst, src, Rc, CC

Aode .	Version	Mnemonic and Form CPSD Rd↑, Rs↑, Rc, C	Clooks		Clocks	Operation If result of dst<0:15>-src<0:15> meets
R	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc		1 ₁ 0 ₁ 1 ₁ 0	25	CC condition in instruction, then Z flag←1. Rs<0:15>←Rs<0:15>-2
R	S	CPSD RRd1, RRs1, Rd 1,0,1,1,1,1,0,1,1 0,0,0,0 Rc	7	1 ₁ 0 ₁ 1 ₁ 0 CC	25	Rd<0:15>←Rd<0:15>-2 Rc<0:15>←Rc<0:15>-1
					0 = 8F = 0	
				B9FI	0 a sAR 0	Description
					0 0,0,0 0RESS	The source word operand is compared to the destination word operand. Both the source and destination operands are words in memory addressed by the
					70 0 0	general-purpose registers designated in the Rd and Rs (or RRd and RRs) fields of the instruction. The comparison is achieved by subtraction. The contents of
					1329-	the general-purpose register designated by the Rc field of the instruction are decremented by one. The source and destination operands are unaltered. The contents of the Rs and Rd registers are
					O Ax ≠ 0 DRESS	decremented by two. R0 can be designated as the general-purpose source or destination register.
					0 4 xA [8	
					O EX + O	

Flags

C	Z	S	P/V	DA	H
	*	_	*	_	-

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise. P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPSDB

COMPARE byte strings in memory, autodecrement

CPSDB

CPSDB dst, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks	Operation September 1997
		CPSDB Rd↑, Rs↑, Rc, CC		If result of dst<0:7>-src<0:7> meets
IR	Sinerii netra	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 Rs 1 ₁ 0	11,0	CC condition in instruction, then Z flag←1.
In	NS	0101010 Rc Rd C	C 25	Rs<0:15>←Rs<0:15>-1
		CONTRACTOR DE LA CONTRA		Rd<0:15>←Rd<0:15>-1
		CPSDB RRd1, RRs1, Rc, CC	70,05	Rc<0:15>←Rc<0:15>-1
IR	e mor	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 RRs 1 ₁ 0	11,0	the Outstate Outstand
in	S	0,0,0,0 Rc RRd C	C 25	5R [0.0,0,0]

Description

The source byte operand is compared to the destination byte operand. Both the source and destination operands are bytes in memory addressed by the general-purpose registers designated in the Rd and Rs (or RRd and RRs) fields of the instruction. The comparison is achieved by subtraction. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by one. The source and destination operands are unaltered.

R0 can be designated as the general-purpose source or destination register.

Flags



Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPSDR

COMPARE word strings in memory, autodecrement and repeat

CPSDR

CPSDR dst, src, Rc, CC

Mode	Version	Mnemonic and Form CPSDR Rd↑, Rs↑, Rc, Co	cicala C		Clocks
IR	NS .	1,0,1,1,1,0,1,1 0,0,0,0 Rc	Rs Rd	1,1,1,0 CC	11 + 14n*
IR	1-<8 1-<8	CPSDR RRdî, RRsî, Rc	, CC RRs RRd	1 ₁ 1 ₁ 1 ₁ 0	11 + 14n*

^{*}n is the number of iterations.

Operation

If result of dst<0:15>-src<0:15> meets CC condition in instruction, then Z flag \leftarrow 1. Rs<0:15> \leftarrow Rs<0:15>-2 Rd<0:15> \leftarrow Rd<0:15>-2

Rc<0:15>←Rc<0:15>-2

Repeat until termination.

Description

The source word operand is compared to the destination word operand. Both the source and destination operands are words in memory addressed by the general-purpose registers designated in the Rd and Rs (or RRd and RRs) fields of the instruction. The comparison is achieved by subtraction. Both source and destination operands are unaltered. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of the Rs and Rd registers are both decremented by two. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C Z S P/V DA H

P/V DA H Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPSDRB

COMPARE byte strings in memory, autodecrement and repeat

CPSDRB

CPSDRB dst, src, Rc, CC

Mode	Version	Mnemonic and Form CPSDRB Rd↑, Rs↑, Rc, 0	CC		Clocks
IR	NS	1,0,1,1,1,0,1,0	Rs	1,1,1,0	11 + 14n*
in	NS.	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd	CC	11 + 14n*
		CPSDRB RRd1, RRs1, R	lc, CC		
IR	6	1,0,1,1,1,0,1,0	RRs	1,1,1,0	11 + 14n*
11.1	3	0,0,0,0 Rc	RRd	CC	11 + 1411

^{*}n is the number of iterations.

Operation

If result of dst<0:7>-src<0:7> meets CC condition in instruction, then Z flag \leftarrow 1. Rs<0:15> \leftarrow Rs<0:15> \leftarrow 1 Rd<0:15> \leftarrow Rd<0:15>-1

Rc<0:15>←Rc<0:15>-1

Repeat until termination.

Description

The source byte operand is compared to the destination byte operand. Both the source and destination operands are bytes in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of the Rs and Rd registers are both decremented by one. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C Z S P/V DA H

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise. P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPSI

COMPARE word strings in memory, autoincrement

CPSI dst, src, Rc, CC

CPSI

Mode	Version	milionio una i omi			Clocks
IR III	School meter	1,0,1,1,1,0,1,1	Rs	0,0,1,0	05
III	NS	0101010 Rc	Rd	CC	25
		CPSI RRd1, RRs1, Rc, C	C		
IR	ion.	1,0,1,1,1,0,1,1	RRs	0,0,1,0	25
III	S	0 ₁ 0 ₁ 0 ₁ 0 Rc	RRd	CC	25

Operation Application

If result of dst<0:15>-src<0:15> meets CC condition in instruction, then Z flag \leftarrow 1. Rs<0:15> \leftarrow Rs<0:15>+2 Rd<0:15> \leftarrow Rd<0:15>+2 Rc<0:15> \leftarrow Rc<0:15>-1

Description

The source word operand is compared to the destination word operand. Both the source and destination operands are words in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The source and destination operands are unaltered. The contents of the Rs and Rd registers are incremented by two.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise. P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

CPSIB

COMPARE byte strings in memory, autoincrement

CPSIB

CPSIB dst, src, Rc, CC

Mode	Version	Mnemonic and Form CPSIB Rd↑, Rs↑, Rc, CC			Clocks	Operation If dst<0:7>-src<0:7> meets CC
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rd	0 ₁ 0 ₁ 1 ₁ 0 CC	25	condition in instruction, then Z flag←1. Rs<0:15>←Rs<0:15>+1
		CPSIB RRdf, RRsf, Rc, C	CC		8a. 00	Rd<0:15>←Rd<0:15>+1 Rc<0:15>←Rc<0:15>-1
IR	not	1,0,1,1,1,0,1,0	RRs	0,0,1,0	25	1.0.111.1.0.1
in	5	0,0,0,0 Rc	RRd	CC	b94	0 0 0 0 P

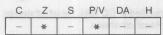
Description

"n le tite mumber of iterations

The source byte operand is compared to the destination byte operand by subtraction. Both the source and destination operands are bytes in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. Both source and destination operands are unaltered. The contents of the Rs and Rd registers are incremented by one.

R0 can be designated as the general-purpose source or destination register.

Flags



Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise. P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPSIR

COMPARE word strings in memory, autoincrement and repeat

CPSIR

CPSIR dst, src, Rc, CC

Mode	Version	Mnemonic and Form CPSIR Rd↑, Rs↑, Rc, CC	Clocks	Operation If result of dst<0:15>-src<0:15> meets
IR	NS	1,0,1,1,1,0,1,1 Rs 0,1,1,0 0,0,0,0 Rc Rd CC	11 + 14n°	CC condition in instruction, then Z flag←1 Rs<0:15>←Rs<0:15>+2 Rd<0:15>←Rd<0:15>+2
IR	S	CPSIR RRd↑, RRs↑, Rc, CC 1 0 1 1 1 1 0 1 1 RRs	11 + 14n*	Rc<0:15>←Rc<0:15> - 1 Repeat until termination.

^{*}n is the number of iterations.

Description

The source word operand is compared to the destination word operand. Both the source and destination operands are words in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. Both source and destination operands are unaltered. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of the Rs and Rd registers are both incremented by two. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

CPSIRB

COMPARE byte strings in memory, autoincrement and repeat

CPSIRB

CPSIRB dst, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks	Operation
		CPSIRB Rd↑, Rs↑, Rc, CC		If dst<0:7>-src<0:7> meets CC
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 Rs 0 ₁ 1 ₁ 1 ₁ 0	1 + 14n°	condition in instruction, then Z flag←
in	INS	0 ₁ 0 ₁ 0 ₁ 0 Rc Rd CC	1 + 1411	Rs<0:15>←Rs<0:15>+1
				Rd<0:15>←Rd<0:15>+1
		CPSIRB RRd1, RRs1, Rc, CC		Rc<0:15>←Rc<0:15>-1
IR	S	1,0,1,1,1,0,1,0 RRs 0,1,1,0	1 + 14n*	Repeat until termination.
11.1	3	0,0,0,0 Rc RRd CC	1 + 1411	

Description

The source byte operand is compared to the destination byte operand. Both the source and destination operands are bytes in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of the Rs and Rd registers are both incremented by one. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C Z S P/V DA H

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise. P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

DAB

traces be trans DECIMAL ADJUST byte

DAB

DAB Rbd

ode	Version	Mnemonic and Form DAB Rbd			Clocks		ration <0:7>←Rb	BCD<0:7	sbok 7>
}	NS, S	1,0,1,1,0,0,0,0	Rbd	0,0,0,0	5				
				00]		98			
						1sAA .			
				0.1.1.0		10,111 Ba			
						369			
						the nam			

Description

A destination byte register, designated by the Rd field of the instruction, is adjusted by the addition of the BCD operand given in the table below. This instruction converts a byte (binary representation) into a two digit binary coded decimal representation, following an arithmetic operation.

Preceding Arithmetic Operation	C Flag Before DAB	dst<4:7> (Hex)	H Flag Before DAB	dst<0:3> (Hex)	BCD<0:7>	C Flag After DAB
ts of the Re and R	15/00	0-9	0	0-9	00	0
	1 0	0-8	0	A-F	06	0
ADDB	0	0-9	1	0-3	06	0
ADCB	0	A-F	0	0-9	60	1
		9-F	0	A-F	66	1
		A-F	1	0-3	66	1
	sedand	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
	0	0-9	0	0-9	00	0
SUBB	0	0-8	1	6-F	FA	0
SBCB	1	7-F	0	0-9	A0	1
	1	6-F	1	6-F	9A	1

Flags

C Z S P/V DA H

- P/V DA H C: Set or reset according to table.
 - Z: Set to 1 if result is zero. Reset otherwise.
 - S: Set to 1 if the most significant bit of the result is set. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- Conditional see description

DBJNZ

DECREMENT byte register and jump on non-zero DBJNZ Rbc, LAB

DBJNZ

Mode	Version	Mnemonic and Form DBJNZ Rbc, LAB	Clocks		Clocks	Operation Rbc<0:7>←Rbc<0:7>-1
RA	NS, S	1 ₁ 1 ₁ 1 Rbc	0 DISPL	ACEMENT	911	If Rc<0:7>≠0, then PC←Updated Pc−2x displacement Otherwise PC←Updated PC.
						U.SEC RAT. N. 10.110.110.110.110
						DEC BRatt, N
						Description 030
						The contents of the general-purpose byte register designated by the Rbc field of the instruction are decremented, and if this produces a non-zero result, a jump is
						executed. The jump address is obtained by subtracting the contents of the 7-bit displacement field, which has been left
						shifted (i.e., word aligned) from the contents of the updated program counter (i.e., incremented by two). The resultant
						address is loaded into the program counter and is used as the jump destination. The instruction displacement field is interpreted as a 7-bit unsigned integer. Thus the range of the relative jump is zero to -127 words with respect
					0 * #F] 10	to the updated PC. If the register decrementation produces a zero result, then the contents of the
						program counter are merely updated by incrementing by two.
		expression which is a field n of the instruction of the instruction of the alcower an assembler error.				Assembler Notation The label LAB is an address which is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside of the allowable range produces

Flags

S P/V DA H Flags are not affected. Set to 1 if result is negative. Reset of

– Unaffected

1 = Set

0 = Cleared

DEC

DECREMENT word

DEC dst, N

DEC

Mode	Version	Mnemonic and Form	Clocks	Operation on the state of the
R	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 Rd noA.	92040	RA NS, S [111,11] Rbc
IR	NS	DEC Rd1, N 010110110111 Rd n	11	
IR	S	DEC RRdî, N 0 0 1 0 1 0 1 1 RRd	11	
		DEC LABEL, N		Description
ant to b	NS Brit	0111101101110000 n	13	A value between 1 and 16 is subtracted from the destination operand word and
DA	SSO	DEC LABSSO, N 0 1 1 1 0 1 1 0 1 0 1 0 0 0 n	14	the result is loaded back into the destination. The desired value to be subtracted is specified by the n field, where n = 0 corresponds to value one
DA	SLO	DEC LABEL, N 0 1 1 0 1 1 0 0 0 0	16	and so on, and n = F corresponds to value 16. The destination is determined by the applicable addressing mode. The original contents of the destination are lost.
X	a the jump action deaded a 7-bi 20 spi se of the relati		14	In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.
X	SSO	OF SECIVIENT	14	
	et le atrietox telou viorem SLO	DEC LABEL (Rx), N 0 1 1 1 0 1 1 1 Rx ≠ 0 n 1 SEGMENT	17	
		OFFSET		Assembler Notation The assembler notation N is a numeric expression which is assembled into the bit
				field n of the instruction. The range of N is 1 to 16, and $n = N - 1$. Specifying an N outside of the allowable range produces an assembler error.

Flags

С	Z	S	P/V	DA	Н
-	*	*	*	-	-

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 on arithmetic overflow. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

DECB

DECREMENT byte

DECB

DECB dst. N

Mode	Version	Mnemonic and Form			Clocks
		DECB Rbd, N			
R	NS, S	1,0,1,0,1,0,1,0	Rbd	11 y n0 10	0 0 4
		DEOD DAA N			
IR	NS	DECB Rd↑, N 0	Rd	n	11
		DECB RRd↑, N			
IR	S	010111011101110	RRd	n	11
		DECB LABEL, N			
DA	NS	0,1,1,0,1,0,1,0		n	13
sente	v alla hoga gi intani edi nid	ADDF	RESS		13
		DECB LABSSO, N			
DA	SSO	0 1 1 1 0 1 1 0 1 1 0		n	14
		0 SEGMENT	OFFS	EI	
		DECB LABEL, N			
DA	SLO	0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 0 1 SEGMENT	0,0,0,0,	n ************	16
		metri benebev-no OFF	SET		
		DECB LABEL (Rx), N			
X	NS	0,1,1,0,1,0,1,0		n	14
		ADDF	RESS		14
		DECB LABSSO (Rx), N			
X	SSO	0 1 1 1 0 1 1 0 1 1 0	Rx ≠ 0	n	14
		0 SEGMENT	OFFS	EI	
		DECB LABEL (Rx), N			
X	SLO	0 1 1 1 0 1 1 0 1 1 0 1 SEGMENT	Rx ≠ 0	n	17
		OFF	SET	***********	.,
		is a of earlier or no not			

Description

Operation dst<0:7>←dst<0:7>−N

A value between 1 and 16 is subtracted from the destination byte operand and the result is loaded back into the destination. The desired value to be subtracted is specified by the n field, where n = 0 corresponds to value one and so on, and n = F corresponds to value 16. The destination is determined by the applicable addressing mode. The original contents of the destination are lost. In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is 1 to 16, and n=N-1. Specifying an N outside of the allowable range produces an assembler error.

Flags

С	Z	S	P/V	DA	Н
-	*	*	*	-	

- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 on arithmetic overflow.

5

DI

0 = Cleared

* = Conditional - see description

DISABLE INTERRUPT

DI

DI LIST

Mode Version	Mnemonic and Form DI LIST			Clocks	Operation FCW<11>←		
- NS, S	0,1,1,1,1,1,0,0	0,0,0,0),0,0,V,N	5c 7 i	FCW<11>← FCW<12>←	FCW<11> fo	
				68			
				198	BHUT N	8030	6 6
					Description		
				(0,0,0,6 AEES	reset to zero of the N and N	enables in the depending upo bits within the	on the values e instruction.
				0,0,0,0. HO	causes the re unaltered, and relevant interr	d a value of ze	t enable to be tro causes the be Reset. The
				138	the vectored i	of V in the instru nterrupt enables of N controls the interrupt enab	e le Al
				Rx # 0	Instruction Bit	FCW Bit/#	Assembler Notation
					0 (8) 02 8A	NVIE/11 VIE/12	NVI VI
				Rx + 0 PQ	TUBINATE	10	068
				a te xA			
					Assembler N	lotation	OUR
				1921	list of either o reserved word NVI, VI. Spec	er notation LIS r both of the fo ds, separated ifying an entry resets the FC	bllowing by commas: disables that
F lags C Z S P/V	/ DA H	Flags ar	e not affected			AG VI9	lags C Z S

DIV RRd, src

Mode	Version	Mnemonic and Form	Clocks	Operation amount motars of
-		DIV RRd, Rs		RRd<0:15>←RRd<0:31>/src<0:15>
R	NS, S	1 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 Rs RRd	107	RRd<16:31>←Remainder
		DIV RRd, IM		9WL BORLIVIO
IM	NS, S	0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 RRd	107	0,1,0,1,1,0,0,0
	110, 0	OPERAND	PAND	NS. 6 St. OPE
		DIV RRd, Rs↑		MO 61
R	NS	$0_10_10_11_11_10_11_11$ Rs $\neq 0$ RRd	107	DIVL RQd, Rst
		DIV RRd, RRs↑		NS 0,0,0,1,1,0,11,0
R	S	0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 RRs ≠ 0 RRd	107	THERE FOR IVID
		DIV RRd, LABEL		Description
		0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 RRd		A 32-bit signed integer (dividend) is
DA	NS	ADDRESS	108	contained in a destination register pair
		DIV RRd, LABSSO		designated by the RRd field of the
		0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 RRd		instruction.
DA	SSO	0 SEGMENT OFFSET	109	A 16-bit signed integer source operand
		DIV RRd, LABEL		(divisor) is determined by the applicable addressing mode. Division is performed
		0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 RRd		to obtain a 16-bit quotient and a 16-bit
DA	SLO	1 SEGMENT	0111	remainder.
		Signal Maileup an OFFSET		The quotient is loaded into the least
		DIV RRd, LABEL (Rx)		significant destination register. The remainder is loaded into the most
X and	NS	$0_11_10_11_11_10_11_11 Rx \neq 0$ RRd	100	significant destination register. The
`	INS	ADDRESS BOR	109	source operand is not altered.
		DIV RRd, LABSSO (Rx)		The original contents of the destination
X	SSO	$0_11_10_11_11_10_11_11 \text{ Rx} \neq 0 \text{ RRd}$	109	are lost unless the division operation is aborted. This occurs if the divisor is zero
ESSENT OF	330	0 SEGMENT OFFSET	109	or if the magnitude of the divisor is less
		DIV RRd, LABEL (Rx)		than or equal to the magnitude of the
		$0_11_10_11_11_0_11_1$ Rx $\neq 0$ RRd		high order half of the dividend.
X	SLO	1 SEGMENT	112	The aborted instruction takes less than
		OFFSET		30 clock cycles.

source operand by the applicable sion is performed ent and a 16-bit

into the least register. The to the most register. The altered.

of the destination sion operation is the divisor is zero the divisor is less agnitude of the lividend.

C	Z	S	P/V	DA	Н
*	*	*	*	neglo	15891

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- Set to 1 if the quotient is less than -2^{15} or greater than/equal to 2^{15} Reset otherwise.
 - Z: Set to 1 if either the quotient or divisor is zero. Reset otherwise.S: Set to 1 if quotient is negative. Reset otherwise.

 - P/V: Set to 1 if division is aborted. Reset otherwise.

DIVL

DIVIDE register quadruple by source long word

DIVL

DIVL RQd, src

	Version	Mnemonic and Form	Clocks	Operation RQd<0:31>←RQd<0:63>/src<0:31>
R	NS, S	DIVL RQd, RRs	744	RQd<32:63>←Remainder
		DIVL RQd, IM\$		DIV PR6, M
IM	NS, S	0 0 0 1 1 1 0 1 0 0	744	NS. 8 010101111.011
		DIVL RQd, Rs↑		NS I LO CO LO STATE
IR	NS	$0_10_10_11_11_10_11_10$ Rs $\neq 0$ RQd	744	DOV BRIG 28e1
		DIVL RQd, RRs1 TOT DEF		2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
R	S	$0_10_10_11_11_10_11_10$ RRs $\neq 0$ RQd	744	Description A Ma
		DIVL RQd, LABEL 0 1 0 1 1 0 0 0 0 0		A 64-bit signed integer (dividend) is con-
DA	NS	ADDRESS	745	tained in a quadruple destination register designated by the RQd field of the
		DIVL RQd, LABSSO		instruction.
DA	SSO	0 1 1 0 1 1 1 0 1 0 0 0 0 0 0 0 RQd 0 SEGMENT OFFSET	746	A 32-bit signed integer source operand (divisor) is determined by the applicable addressing mode. Division is performed
		DIVL AQQ, LABEL		to obtain a 32-bit quotient and a 32-bit remainder.
1	SLO	melandest medime OFFSET	748	The quotient is loaded into the least significant destination register pair. The
		DIVL RQd, LABEL (Rx)		remainder is loaded into the most significant destination register pair. The
X	NS	0 1 1 0 1 1 1 1 0 1 1 0 Rx ≠ 0 RQd ADDRESS	746	source operand is not altered.
		DIVL RQd, LABSSO (Rx)		The original contents of the destination are lost unless the division operation is
X - 30 2	SSO	0 11011110110 Rx ≠ 0 RQd 0 SEGMENT OFFSET	746	aborted. This occurs if the divisor is zero or if the magnitude of the divisor is less than or equal to the magnitude of the
		DIVL RQd, LABEL (Rx)		high order half of the dividend. The aborted instruction takes a maximum
X	SLO	0 1 0 1 1 0 1 1 0 0 Rx ≠ 0 RQd 1 SEGMENT	749	of 60 clock cycles.



- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Set to 1 if the quotient is less than -2^{31} or greater than/equal to 2^{31} . Reset otherwise. 2: Set to 1 if either the quotient or divisor is zero. Reset otherwise.
- S: Set to 1 if quotient is negative. Reset otherwise.
- P/V: Set to 1 if division is aborted. Reset otherwise.

DJNZ

DECREMENT word register and jump on non-zero

DJNZ

DJNZ Rc, LAB

Mode Version Mnemonic and Form DJNZ Rc, LAB RA NS, S 1,1,1,1 Rc	Clocks 1 DISPLACEMENT 11	Operation Rc<0:15>←Rc<0:15>−1 If Rc<0:15>≠0, then PC←Updated Pc−2x displacement. Otherwise PC←Updated PC.
		Description The contents of the general-purpose word register designated by the Rc field of the instruction are decremented and if this produces a non-zero result, a jump is executed. The jump address is obtained by subtracting the contents of the 7-bit instruction displacement field which has been left shifted (i.e., word aligned) from the contents of the updated program counter (i.e., incremented by two). The resultant address is loaded into the program counter and is used as the jump destination. The displacement field is interpreted as a 7-bit unsigned integer. Thus the range of the relative jump is zero to -127 words with respect to the updated PC. If the register decrementation produces a zero result, then the contents of the program counter are merely updated by incrementing by two.
		Assembler Notation The label LAB is an address which is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

El

CIESCRION TO ENABLE INTERRUPT BRODE

SEL

EI LIST

			This is a SYSTEM in	Struction.			
Mode	NS, S	Mnemonic and Form EI LIST 0 1 1 1 1 1 1 1 0 0	0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ V ₁ N	Clocks	Operation FCW<11>←1 FCW<11>←F FCW<12>←1 FCW<12>←F	for N=0 FCW<11> for for V=0	N=1
					Description The interrupt e	enables in the	FCW are se
					to one depend N and V bits w value of one ir the relevant in unaltered, and relevant intern bit designated controls the ve and the bit des non-vectored i	within the instruction these bit posterrupt enable a value of zeupt enable to V in the instructored interrusignated N co	uction. A sitions cause: to be ro causes the causes the uction pt enable bit.
					Instruction Bit	FCW Bit/#	Assembler Notation
					0	NVIE/11 VIE/12	NVI
					Assembler No		
					The assemble list of either or reserved word NVI, VI. Speci interrupt (i.e., s	both of the fo s, separated b fying an entry	llowing by commas: enables that

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

EXCHANGE source word with destination word

EX

EX Rd, src

Mode	Version	Mnemonic and Form EX Rd, Rs	Clocks	Operation src<0:15>↔Rd<0:15>	
R	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₀ 1 Rs Rd	6 0	ольтоттот ези	
IR	NS	EX Rd, Rs↑	1 40	EXB Rod, Pat	
ın	NS		12	0,1,1,0,1,0,0) sn	
IR	S	EX Rd, RRs↑	12	TaRia, Bod and a second	
111	3		12		
DA	NS	EX Rd, LABEL 0 1 1 0 1 0 0 0 0 Rd ADDRESS	15	Description The contents of the source operan are exchanged with the contents of	
DA	SSO	EX Rd, LABSSO 0 1 1 1 0 1 1 0 0 0 0 Rd 0 SEGMENT OFFSET	16	destination operand word. The destination operand is always a general-purpose word register des by the Rd field of the instruction. To	he
DA	SLO	EX Rd, LABEL 0	18	source operand is determined by t appropriate addressing mode. In the IR mode R0 (or RR0) can be designated as the general-purpose register.	A.E
X	NS	EX Rd, LABEL (Rx) 0 1 1 1 0 1 1 1 0 1 Rx ≠ 0 Rd ADDRESS	16	EYB Rbd, LABEL(R) 0.11.10.11.10 NS	
X	SSO	EX Rd, LABSSO (Rx) 0 1 1 1 0 1 1 1 0 1	16	EXB Rbd, LABSSO (
		EX Rd, LABEL (Rx) 0 1 1 0 1 1 0 1 Rx ≠ 0 Rd	0 2 28 16	EXERBO, LABEL (R)	
X	SLO	1 SEGMENT	19	BLO I SECMENT	



C Z S P/V DA H

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

EXB

* = Conditional - see description

EXCHANGE source byte with destination byte

EXB

EXB Rbd, src

Mode	Version	Mnemonic and Form	Clocks	Operation
		EXB Rbd, Rbs		src<0:7> ↔ Rbd<0:7>
R	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ ·1 ₁ 0 ₁ 0 Rbs Rbd	8 A 6	R NS.S TIOTEOTET
IR	110	EXB Rbd, Rs↑	10	EX Rd, Rat
IH	NS	0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 Rs Rbd	12	R MS (0,0,1,0,1,1,0)
ID		EXB Rbd, RRs↑	10	fel (8, 69, 8)
IR	S	0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 RRs Rbd	12	91/11/02/1919
		EXB Rbd, LABEL		Description X
	NS	0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd ADDRESS	15	The contents of the source operand byte are exchanged with the contents of the destination operand byte. The destination
	SSO		16	operand is always a general-purpose byte pregister designated by the Rbd field of the instruction. The source operand is deter-
DA	SLO	EXB Rbd, LABEL 0 1 1 0 0 0 0 0 0 Rbd 1 SEGMENT SEGME	18	mined by the appropriate addressing mode. In the IR mode R0 (or RR0) can be designated as the general-purpose source register.
X	NS	EXB Rbd, LABEL (Rx) 0	16	EX Ed. LABEL (EX) (0,10,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,
X	SSO	EXB Rbd, LABSSO (Rx) 0 1 1 0 1 1 0 0 Rx ≠ 0 Rbd 0 SEGMENT OFFSET	16	EX Rd, LABSSO (EX 0,1,1,0 0,1,1,0 0,1 0,1,1,0 0,1 0,1 0,1
×	SLO	EXB Rbd, LABEL (Rx) 0 1 1 1 0 1 1 1 0 0 Rx ≠ 0 Rbd 1 SEGMENT OFFSET	19	X SLO 1 SEGMENT
Flags C Z	S P/V	DA H Flags are not affected.		

EXTS

EXTEND sign of a word

EXTS RRd

EXTS

Mode Version	Mnemonic and Form	Clocks Operation If RRd<0:15>is negative,
R NS, S		RRd<16:31>←1's; otherwise RRd<16:31>←0.
		Description The destination is a general-purpose register pair, designated by the RRd field
	significant byte of the first cach position of byte. In this manner, byte. In this manner, operand is preserve extended from eight	of the instruction. The sign bit of the less significant register of the pair is copied into each bit position of the most significant register. In this manner, the sign of the operand is preserved as the operand is extended from 16 to 32 bits in length.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

EXTSB

EXTEND sign of a byte

EXTSB

Mode	Version	Mnemonic and Form		Clocks	Operation If Rd<0:7>is negative,	
R	NS, S	1+0+1+1+0+0+0+1 36(3-b)(1-0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1	Rd 0,0,0,0	6R11 1	Rd<8:15>←1's; otherwise Rd<8:15>←0.	

Description

The destination is a general-purpose register, designated by the Rd field of the instruction. The sign bit of the less significant byte of the register is copied into each position of the most significant byte. In this manner, the sign of the operand is preserved as the operand is extended from eight to 16 bits.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

EXTSL

EXTEND sign of a long word

EXTSL

EXTSL RQd

Mode	Version	Mnemonic and Form			Clocks
R	NS, S	EXTSL RQd 1	RQd	0,1,1,1	010

Operation

If RQd<0:31>is negative, RQd<32:63> \leftarrow 1's; otherwise RQd<32:63> \leftarrow 0.

Description

The destination is a general-purpose register quad, designated by the RQd field of the instruction. The sign bit of the less significant register pair of the quad is copied into each bit position of the most significant register pair. In this manner, the sign of the operand is preserved as the operand is extended from 32 to 64 bits.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

HALT

* = Conditional - see description

EXTEND TIAH a tong word

HALT

HALT

			This is a SYSTEM in	struction.		
Mode R	Version NS, S	*Interrupts are r	occognized at the ree cycle period.	Clocks 8 + 3n*	Operation	Micce Version
					Description Instruction execution CPU will be in a halte interrupt or reset is re While in the halted st be acknowledged and continue.	ed state until an eceived.
Flags C Z	S P/\	/ DA H	Flags are not affected	Flogs an	A AO	Plags C Z S Ph
- - - = Una 1 = Set 0 = Clea	ffected					

INPUT word to register from I/O port

IN

IN Rd, src

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form IN Rd, Rp			Clocks	Operation Rd<0:7>←port src<0:7>
PR	NS, S	0,0,1,1,1,1,0,1	Rp	Rd	10	0,0,1,1,1,1,0,0
		IN Rd, PORT		T		IN Rod, PORT
PA	NS, S	0,0,1,1,1,0,1,1 PORT AD	Rd DRESS	0,1,0,0	12	0,1,0,1,1,1,0,0 A TRO9

Description

A general-purpose byte destination register designated by the Rd field of the instruction is loaded from an input port. The port address source is determined by the applicable port addressing mode. The original contents of the destination are lost.

R0 can be designated as the general-purpose port source register.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

INB

INPUT byte to register from I/O port

INB

IN Rbd, src

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form			Clocks	Operation Rbd<0:7>←port s	
PR	NS, S	0,0,1,1,1,1,0,0	Rp	Rbd	10	1.0,1,1,1,1,0,	
		IN Rbd, PORT				Rd, POR	
PA	NS, S	0,0,1,1,1,0,1,0	Rbd	0111010	12	titio, tititio,	
171	140, 0	PORT ADI	DRESS		DDRESS	A TROP	

Description

A general-purpose byte destination register designated by the Rbd field of the instruction is loaded from an input port. The port address source is determined by the applicable port addressing mode. The original contents of the destination are lost.

R0 can be designated as the generalpurpose port source register.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

INCREMENT word

INC

INC dst, N

	Version	Mnemonic and Form INC Rd, N	Clocks	Operation dst<0:15>←dst<0:15>+N		
R	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 Rd n	hd 4 0	01011011011 8 8N 11011011011		
IR	NS	INC Rdî, N 0 0 1 0 1 0 0 1 Rd Rd Rd	b.11 Q	NCB Rdj. N (010,11,0,1,0,0)		
IR	S	INC RRd↑, N 0 0 1 0 1 0 0 1 RRd n	h911 To	INCB RRdf, N		
		INC LABEL, N		Description		
DA	NS	0 1 1 0 1 0 0 1 0 0 0 n ADDRESS	13	A value between 1 and 16 is added to the destination operand word and the result is loaded back into the destination. The desired value to be added is specified by the n field, where n = zero corresponds to value one and so on, and n = F		
	SSO	INC LABSSO, N 0 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0	14			
DA	SLO	INC LABEL, N 0 1 1 1 0 1 0 0 0 0 n 1 SEGMENT OFFSET	16	corresponds to value 16. The destination is determined by the applicable addressing mode. The original contents of the destination are lost.		
	NS		14	In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.		
	SSO	INC LABSSO (Rx), N 0 1 1 1 0 1 1 0 1 1 Rx ≠ 0 n 0 SEGMENT OFFSET	14	Assembler Notation The assembler notation N is a numeric expression which is assembled into the b		
X Soldon SLO		INC LABEL (Rx), N 0 1 1 1 0 1 0 1 Rx ≠ 0 n 1 SEGMENT OFFSET	0 A HI 10	field n of the instruction. The range of N one to 16, and n = N - 1. Specifying a N outside of the allowable range produ an assembler error.		

Flags

C Z S P/V DA H Secretary Set to 1 if result is zero. Reset otherwise. * *

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 on arithmetic overflow. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

INCB

INCREMENT byte

INCB

INCB dst, N

R		Chicago and the Control of the Contr	Clocks	Operation and addition about			
	NS, S	INCB Rbd, N 1 0 1 0 1 0 0 0 Rbd	b84	dst<0:7>←dst<0:7>+N			
R	NS	INCB Rd↑, N 0 0 0 1 1 0 1 0 0 0 Rd	b91 I	INC R41, N			
П	INO	INCB RRd↑, N		INC RRAT. N			
R	S	0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 RRd n	b811	0,0,1,0,1,0,0 B 8 A			
		INCB LABEL, N		Description OM			
	NS	0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁	13	A value between 1 and 16 is added to the destination operand byte and the result is			
	SSO	NCB LABSSO, N	14	loaded back into the destination. The desired value to be added is specified by the n field, where n = zero corresponds to one and so on, and n = F corresponds to			
odnien s o		INCB LABEL, N 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	16	value 16. The destination is determined by the applicable addressing mode. The original contents of the destination are lost.			
	or R(30), ca s-purpose NS	INCB LABEL (Rx), N 0 1 1 1 0 1 0 0 Rx ≠ 0 n ADDRESS	14	In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.			
		INCB LABSSO (Rx), N		Assembler Notation			
numeric X	SSO	0 1 1 1 1 0 1 0 0 0 Rx ≠ 0 n 0 SEGMENT OFFSET	14	The assembler notation N is a numeric expression which is assembled into the bit			
nge of N is oilying an e produces		INCB LABEL (Rx), N 0 1 1 1 1 0 1 1 0 1 0 1 0 Rx ≠ 0 n		field \vec{n} of the instruction. The range of N is one to 16, and $\vec{n} = N - 1$. Specifying an N outside of the allowable range produces			
<	SLO	1 SEGMENT OFFSET	17	an assembler error.			

Flags

С	Z	S	P/V	DA	Н
	4	4		0	4

= Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 on arithmetic overflow. Reset otherwise.

DA: Set to 0.

H: Set on carry from least significant digit. Reset otherwise.

INPUT word from I/O port to memory, autodecrement

IND

IND dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form IND Rd↑, Rp, Rc	Clocks	Operation dst<0:15>←port src<0:15>	
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₀ 1 ₁ 1 Rp 1 0 ₁ 0 ₁ 0 ₁ 0 Rc Rd 1	101010 21	Rd<0:15>←Rd<0:15>-2 Rc<0:15>←Rc<0:15>-1	
IR, PR	S	IND RRd↑, Rp, Rc 0 0 1 1 1 0 1 1 Rp 1 0 0 0 0 Rc RRd 1	101010 101010 21	NOB RR87, Ap. Rc. Rc 0.00 0.00 Rc	

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then decremented by two. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose port source or destination register.

Flags

C Z S P/V DA H

P/V: Set to 1 if result of decrementing Rc register is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

INDB

INPUT byte from I/O port to memory, autodecrement

INDB

INDB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form INDB Rd↑, Rp, Rc			Clocks	Operation dst<0:7>←port src<0:7>		
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 Rp 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc Rd 1 ₁ 0 ₁ 0 ₁ 0	1,0,0,0	21	Rd<0:15>←Rd<0:15>-1 Rc<0:15>←Rc<0:15>-1		
		INDB RRd↑, Rp, Rc	Rp	1,0,0,0				
IR, PR	S	0 ₁ 0 ₁ 0 ₁ 0 Rc	RRd	1,0,0,0	21	H, PH S [01810, 1] Bc		
						Description		
						Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose registers designated by Rd and Rc are then decremented by one. This instruction uses both indirect register memory addressing and port register port addressing modes.		
						R0 can be designated as the general- purpose port source or destination register.		
						тедізісі.		

Flags

C Z S P/V DA H

C Z S P/V DA H P/V: Set to 1 if result of decrementing Rc register is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

INPUT word from I/O port to memory, autodecrement and repeat

INDR

INDR dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form INDR Rd↑, Rp, Rc		Clocks
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rp Rd	$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$
IR, PR	S	INDR RRd↑, Rp, Rc 0 0 1 1 1 0 1 1 0 0 0 0 Rc	Rp RRd	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$

*n is the number of iterations.

Operation Application

dst<0:15>←port src<0:15> Rd<0:15>←Rd<0:15>-2 Rc<0:15>←Rc<0:15>-1 Repeat until termination.

nepeat until termination.

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose registers designated by Rd are then decremented by two. The contents of the general-purpose register designated by Rc are decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose port source or destination register.

Flags

C Z S P/V DA H

P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

INDRB

INPUT byte from I/O port to memory, autodecrement and repeat

INDRB

INDRB dst, Rp, Rc

is is a CVCTEM instruction

Mode	Version	Mnemonic and Form			Clocks	Operation
		INDRB Rd↑, Rp, Rc				dst<0:7>← port src<0:7>
R, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rp Rd	0,0,0,0	11 + 10n*	Rd<0:15>←Rd<0:15>−1 Rc<0:15>←Rc<0:15>−1
		Ranger until termination		10101010		Repeat until termination.
		INDRB RRd1, Rp, Rc	Rp	11.0.0.0		
IR, PR	S	0 ₁ 0 ₁ 0 ₁ 0 Rc	RRd	0,0,0,0	11 + 10n*	
		*n is the numbe	r of iteration	ns.	olianeti lo te	
						Description
						Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into the memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The
						contents of the general-purpose register designated by Rd is then decremented by one. The contents of the general-purpose register designated by Rc are decremented by one. The instruction is terminated when the result of this decrementation reaches zero. This instruction is interruptible.
						designated by Rd is then decremented by one. The contents of the general-purpose register designated by Rc are decremented by one. The instruction is terminated when the result of this decrementation reaches zero. This instruction is interruptible. This instruction uses both indirect register
						designated by Rd is then decremented by one. The contents of the general-purpose register designated by Rc are decremented by one. The instruction is terminated when the result of this decrementation reaches zero. This instruction is interruptible.

Flags

P/V DA H

P/V: Set to 1.

- = Unaffected
- 1 = Set 0 = Cleared
- * = Conditional see description

INPUT word from I/O port to memory, autoincrement

INI

INI dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form INI Rd↑, Rp, Rc			Clocks
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rp Rd	0,0,0,0	21
IR, PR	S	INI RRd1, Rp, Rc 0 1 0 1 1 1 1 1 0 1 1 1 0 1 0 1 0 1 0 Rc	Rp RRd	0,0,0,0	21

Operation _____

dst<0:15>←port src<0:15> Rd<0:15>←Rd<0:15>+2 Rc<0:15>←Rc<0:15>-1

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose port source or destination register.

Flags

C Z S PV DA H

P/V: Set to 1 if result of decrementing Rc register is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

INIB

INPUT byte from I/O port to memory, autoincrement

INIB

INIB dst, Rp, Rc

This is a SYSTEM instruction.

R, PR NS	INIB Rd↑, Rp, Rc	NS NIB Rd↑, Rp, Rc		Clocks 21	Operation dst<0:7>←port src<0:7> Rd<0:15>←Rd<0:15>+1 Rc<0:15>←Rc<0:15>-1
R, PR s	INIB RRd1, Rp, Rc 0,0,1,1,1,1,0,1,0 0,0,0,0 Rc	Rp RRd	0101010	21	1N1 SRd1, Ro, Rc 0,015,11,10,11 2, PR S 0,010,01 Ro
					Description
					Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose registers designated by Rd are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one.
					This instruction uses both indirect register memory addressing and port register port addressing modes.
					R0 can be designated as the general- purpose port source or destination register.

Flags

C Z S P/V DA H

C Z S P/V DA H P/V: Set to 1 if result of decrementing Rc register is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

INPUT word from I/O port to memory, autoincrement and repeat

INIR

INIR dst. Rp. Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form		Clocks	Operation	
		INIR Rd↑, Rp, Rc			dst<0:15>←port src<0:15>	
IR. PR	NS	0,0,1,1,1,0,1,1	Rp	0,0,0,0	Rd<0:15>←Rd<0:15>+2	
in, rn	INS	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd	$\begin{array}{ c c c c c c c c c c c c c c c c c c c$		
		INIR RRd1, Rp, Rc			Repeat until termination.	
IR. PR	6	010111110111	Rp	$0_10_10_10_1 + 100$		
in, rn	3	0101010 Rc	RRd	0101010 11 + 1011	1 38 0,0,0,0	

*n is the number of iterations.

Description

"n is the number of iterations.

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

The instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose port source or destination register.

Flags

P/V DA 1

P/V: Set to 1.

- = Unaffected

1 ` = Set

0 = Cleared

INIRB

INPUT byte from I/O port to memory, autoincrement and repeat

INIRB

INIRB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form		Clocks	Operation	
		INIRB Rd↑, Rp, Rc			dst<0:7>←port src<0:7>	
IR. PR	NC	0,0,1,1,1,0,1,0	Rp	0101010 11 100*	Rd<0:15>←Rd<0:15>+1	
In, Fh	NS	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd	$0_10_10_10_1$ 11 + 10n*	Rc<0:15>←Rc<0:15>-1	
		INIRB RRd↑, Rp, Rc			Repeat until termination.	
IR. PR	0	0,0,1,1,1,0,1,0	Rp	0 ₁ 0 ₁ 0 ₁ 0 11 + 10n*		
In, rn	5	0101010 Rc	RRd	0101010 11 + 100		

^{*}n is the number of iterations.

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one. This instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose port source or destination register.

Flags

C Z S P/V DA H

P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

RETURN from interrupt

Description

IRET

This instruction causes a return from an interrupt or trap. The program status that was pushed on the system stack is popped to restore the pre-interrupt processor status. The System Stack Pointer contents are modified to reflect the entries that are removed.

Flags

C Z S P/V DA H

* * * * * * *

The flags will be restored to pre-interrupt values.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

5

IRET

JP

JUMP conditional

JP cc, dst

JP

Mode	Version	Mnemonic and Form	Clocks	PC<0:15>←dst<0:15> if condition is met. PC<0:15>←Updated PC if condition failed.
IR	NS	CC True/ JP CC, Rd↑ [0_10_10_11_11_11_10] Rd CC	CC False	
IR	S	JP CC, RRd↑ 0,0,0,1,1,1,1,1,0 RRd CC	15 / 10	
DA	NS	JP CC, LABEL 0 1 0 1 1 1 1 1 0 0	7 / 7	Description The program executes a jump if the condition set in the condition code field of the instruction is met. A destination
DA	SSO	JP CC, LABSSO 0 1 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0 CC 0 SEGMENT OFFSET	8/8	address is obtained according to the applicable addressing mode, and is loaded into the program counter. If the condition set in the condition code of the
DA	SLO	JP CC, LABEL 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 CC 1 SEGMENT OFFSET	10 / 10 S+ <dts< td=""><td>instruction is not met, then the value in the program counter is merely updated. R0 can be designated as the general-purpose destination register Rd or RRd in</td></dts<>	instruction is not met, then the value in the program counter is merely updated. R0 can be designated as the general-purpose destination register Rd or RRd in
X	NS	JP CC, LABEL (Rx) 0 1 0 1 1 1 1 0 Rx ≠ 0 CC ADDRESS	S+<818	the IR mode. Assembler Notation
X	SSO	JP CC, LABSSO (Rx) 0 1 1 0 1 1 1 1 1 1 1 0	8 / 8	Specifying condition CC is optional. If none is specified, the CC field of the instruction is set to hex eight.
X	SLO	JP CC, LABEL (Rx) 0 1 1 0 1 1 1 1 1 1 1 0	11 / 11	
		OFFSET		

P/V DA H Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

JR

- = Unaffected 1 = Set 0 = Cleared

* = Conditional - see description

JUMP conditional relative

JR CC, LAB

JR

RA	Version	JR CC, LAB	tab	Clocks	LACEMENT	Clocks 6	Operation PC←Updated PC+2x displacement if condition is met:	
n.A	NS, S	1,1,1,0	CC	DISPLACEMENT			otherwise PC←Updated PC.	
						DR II	1.D Ref, Rs	
						1 688 1	LD RRoff, Rs	
							ar iasard	
							Description	
							A program jump is taken if the condicode set in the CC field of the instruction is met. If the condition is met, the confit the program counter are updated (incremented by two) and added to the contents of the 8-bit displacement fire	ction ntents he
addressing mode. The contents of the source are unaffered, and the original contents of the destination are lost in the IR mode. R0 (or RIP0) can be designated as the concrui-purpose.					Fla		the instruction, after the latter has be sign extended and left shifted (word aligned). The result is then loaded in program counter as the jump address	een ito the
			Norm orm eriti				the condition is failed, the program counter is merely incremented by two The range of the jump is +127 to words with respect to the updated P The program counter segment numbers.	128 C.
							remains unchanged.	>
							Assembler Notation	
							used by the assembler to generate t	he I PC. ent
		re RA addressing sed by the assen placement relative				[0:0:0] KCEMENT	an assembler error.	AF If
	which is assured						instruction is set to hex eight.	
							NS, S 011 110 011 1 (Rx) (Rx) (Rx)	

5-79

LD

LDR

1 = Set 0 = Cleared

* = Conditional - see description

LOAD word register into memory

LD dst, Rs LDR LAB, Rs LD

LDR

Mode	Version	Mnemonic and Form	Clocks	Operation Application Application
				dst<0:15>←Rs<0:15>
				AA NS.S (11111) CC
IR	NS	LD Rd↑, Rs 0 0 1 1 0 1 1 1 1 Rd Rs	7 8	
	140	LD RRd↑, Rs	_ ~	
IR	S	0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 RRd Rs	8	
		LD LABEL, Rs	7	Description
DA	NS	0,1,1,0,1,1,1,1,0,0,0,0 Rs ADDRESS	11	The word contents of the source register are loaded into the word destination. The
			7	source operand is always a general-
DA	SSO TELE	0 1,1,1,0,1,1,1,1,0,0,0,0,0 Rs 0 SEGMENT OFFSET	12	purpose word register designated by the Rs field of the instruction. The destination is determined by the applicable
noed	memaosiqaib add redal edd SLO	LD LABEL, RS 0 1 1 1 0 1 1 1 1 0 0 0 0 0 Rs 1 SEGMENT	14 8 14	addressing mode. The contents of the source are unaltered, and the original contents of the destination are lost.
	betreet nertret the jump soon duthe product	ups) and (bangia OFFSET		In the IR mode, R0 (or RR0) can be designated as the general-purpose
X	NS		12	destination register. Note: In the BA and BX addressing modes the segmented version requires
	SSO	LD LABSSO (Rx), Rs 0 1 1 1 1 0 1 1 1 1 1 1 Rx ≠ 0 Rs 0 SEGMENT OFFSET	12	the designation of a register pair, RRd ≠ 0, as destination base address register.
, elin		LD LABEL (Rx), Rs 0 1 1 1 1 0 1 1 1 1 1 1 Rx ≠ 0 Rs		Assembler Notation
	SLO	1 SEGMENT OFFSET	15	In the RA addressing mode the assembled displacement is a signed two's complement number with a range of
		LDR LAB, Rs 0,0,1,1,1,0,0,1,1,1,0,0,0,0,0 Rs	٦	+32,767 to -32,768.
	NS, S	DISPLACEMENT	14	In the RA addressing mode the label LAB is used by the assembler to generate the displacement relative to the updated PC.
ВА	NS, S (See note)	LD Rd↑ (D), Rs 0 0 1 1 1 0 0 Rs DISPLACEMENT	14	In the BA addressing mode the value D is an unsigned integer which is assembled
вх	NS, S (See note)	LD Rd↑ (Rx), Rs 0 1 1 1 0 0 1 1 Rd ≠ 0 Rs Rx	14	into the binary displacement. A LAB or D which results in a displacement outside the allowable range produces an assembler error.
Flags				Flags
	Z S P/V	DA H Flags are not affect	ed.	

LOAD word into register LD LD LD Rd, src LDR Rd, LAB LDR LDR Operation Mode Version Mnemonic and Form Clocks LD Rd. Rs Rd<0:15>←src<0:15> NS, S 1,0,1,0,0,0,0,1 R Rs Rd 3 LD Rd. IM 0,0,1,0,0,0,0,1 0,0,0,0 Rd IM NS. S **OPERAND** LD Rd, Rs↑ IR NS 0,0,1,0,0,0,0,1 $Rs \neq 0$ Rd IR S $0_10_11_10_10_10_10_11$ RRs $\neq 0$ Description LD Rd, LABEL 0,1,1,0,0,0,0,1,0,0,0 Rd The source operand word is loaded into DA 9 NS **ADDRESS** the destination word register. The source operand is determined by the applicable LD Rd, LABSSO addressing mode and the destination is 0,1,1,0,0,0,0,1,0,0,0 always a general-purpose register DA SSO 10 SEGMENT **OFFSET** designated by the Rd field of the instruction. The contents of the source LD Rd. LABEL operand are unaltered, and the original 0,1,1,0,0,0,0,1 0,0,0 Rd contents of the destination are lost. DA SEGMENT 12 SLO Note: In the BA and BX addressing **OFFSET** modes the segmented version requires the designation of a register pair, LD Rd, LABEL (Rx) RRs ≠ 0, as source base address $0_11_11_10_10_10_10_11$ Rx $\neq 0$ Rd X NS 10 register. **ADDRESS** LD Rd, LABSSO (Rx) 0,1,1,0,0,0,0,1 Rx ≠ 0 Rd X SSO 10 SEGMENT OFFSET **Assembler Notation** LD Rd, LABEL (Rx) In the RA addressing mode the assembled displacement is a signed two's $0_11_11_10_10_10_10_11$ Rx $\neq 0$ Rd X SLO SEGMENT 13 complement number with a range of +32,767 to -32,768. **OFFSET** In the RA addressing mode the label LAB LDR Rd, LAB is used by the assembler to generate the 0,0,1,1,0,0,0,1 0,0,0,0 Rd displacement relative to the updated PC. RA NS. S 14 DISPLACEMENT in the BA addressing mode the value D is an unsigned integer which is assembled LD Rd, Rs1 (D) into the binary displacement. NS. S $0_10_11_11_10_10_10_11$ Rs $\neq 0$ Rd BA 14 DISPLACEMENT A LAB or D which results in a displace-(See note) ment outside the allowable range LD Rd, Rs† (Rx) produces an assembler error. NS, S 0,1,1,1,0,0,0,1 Rs ≠ 0 BX 14 (See note) Flags C S P/V DA H Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

LD

LOAD IMMEDIATE word into memory

LD dst, IM

LD

Mode	Version	Mnemonic and Form	Clocks	Operation dst<0:15>←IM<0:15>	
IR	NS	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 Rd 0 ₁ 1 ₁ 0 ₁ 1	11	NS. 8 11010101010101	
		OPERAND		LD Rd, IM	
		LD RRd↑, IM		0,0,0,0,0,0,0	
IR	S	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 RRd 0 ₁ 1 ₁ 0 ₁ 1 OPERAND	HUAR		
		LD LABEL, IM		NS CONTRODUCT	
		0,1,0,0,1,1,0,1 0,0,0,0 0,1,0,1			
DA	NS	ADDRESS OPERAND	14	1.881.68.01	cs.
		OPERAND		Description	2.3
		LD LABSSO, IM		The immediate word operand following	00
	SSO		15	The state of the s	
		OPERAND		the destination. The destination is	
		LD LABEL, IM		determined by the applicable address mode. The original contents of the	ing
		0,1,0,0,1,1,0,1,0,0,0,0,0,1,0,1		destination are lost.	
DA	SLO	1 SEGMENT	17	In the IR mode, R0 (or RR0) can be	
DA	January State	OITOLI	10,0,0	designated as the general-purpose	
		OPERAND.		destination register.	
		LD LABEL (Rx), IM	TERT		
X	NS		15	(xii) LEBAL (BX)	
^	36.11	OPERAND	PESS PESS	010101010101010	
		LD LABSSO (Rx), IM			
		$0_11_10_10_11_11_10_11$ Rx $\neq 0$ $0_11_10_11$		10 BY FY BESO (B9)	
X	SSO	0 SEGMENT OFFSET	15	SSO O SCOMENT	
		OPERAND			
		LD LABEL (Rx), IM		(R) J384J, LNOJ 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0	
		$0 1 0 0 1 1 0 1 $ $Rx \neq 0$ $0 1 0 1 $		SLO IT SLEWENT	
X	SLO	1 SEGMENT OFFSET	18		
		ODEDAND		BA LIS BOJ	
				VISSIG BISSIG	
				LD Rd, Ret (D)	
				NS, S [0:0:1:1:0:0:0:0] (See note) CISPLA	
				(See note) [DISPUA	
				LD Rd. Pat (Ext.	
			U to alta il	NS, S 0, 1, 1, 0, 0, 0, 0, 0	XE
Flags					
C 2	Z S P/\	/ DA H Flags are not affected.			
		e not affected			
- = Una	affected	Wasans Ion 9			
1 = Set					

LDA

LDAR

- = Unaffected1 = Set0 = Cleared

* = Conditional - see description

LOAD address to register

LDA dst, addr LDAR dst, LAB LDA

LDAR

Mode	Version	Mnemonic and Form		Clocks	Operation dst←address of source
					DASIO DISPLICATION SIN AF
					10AR RR4 1AB 0.0.111.01101 1A 8 08FL
	NS binarago soni	LDA Rd, LABEL 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	Rd	12	Description The address of a source operand is determined from the applicable addressing mode. This address is then loaded into the general-purpose destination register designated by the Rd (or RRd) field of the instruction. The
DA 19	SSO	0 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0	RRd SET	13	original contents of the destination are lost.
	SLO	LDA RRd, LABEL 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	RRd	15	Note: The destination will be a word operand in the non-segmented version and requires the designation of a general-purpose word register, Rd. The destination will be a long word operand in the segmented version and requires the
×	NS	LDA Rd, LABEL (Rx) $ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	Rd	13	designation of a general-purpose register pair, RRd. In the BA and BX addressing modes, the source base address register will be a
X	SSO	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	RRd SET	13	general-purpose register, Rs, in the non-segmented version, and a register pair, RRs, in the segmented version.
AB is a the	SLO 1991	LDA RRd, LABEL (Rx) 0	RRd	16	Assembier Notation In the RA addressing mode the assembled displacement is a signed two's complement number with a range of
RA	NS, S (See note)	0_0_1_1_1_0_1_0_0 0_0_0_0 DISPLACEMENT	Rd	15	+32,767 to -32,768. In the RA addressing mode the label LAB is used by the assembler to generate the
ВА	NS, S (See note)	LDA Rd, Rs↑ (D) 0	Rd	15	displacement relative to the updated PC. In the BA addressing mode the value D is an unsigned integer which is assembled into the binary displacement.
вх	NS, S (See note)	LDA Rd, Rs↑ (Rx) 0 1 1 1 1 1 0 1 1 0 0 Rs ≠ 0 Rx	Rd	15	A LAB or D which results in a displace- ment outside the allowable range produces an assembler error.

LDAR

LOAD RELATIVE address to register

LDAR dst, LAB

(See also LDA - Load address to register)

LDAR

Mode Version	Mnemonic and Form LDAR Rd, LAB			Clocks	Operation dst←address of source	
RA NS	0,0,1,1,0,1,0,0	O ₁ O ₁ O ₁ O	Rd	15	data dagrees of course	
	LDAR RRd, LAB					
RA s	0,0,1,1,0,1,0,0	O ₁ O ₁ O ₁ O	RRd	15		
		OLIVILIAI				
					130 A 50 AO 4	
					Description	
					Description The address of a source operand is	
					determined by the RA addressing mode.	
					This address is then loaded into the	
					general-purpose destination register designated by the Rd or RRd field of the	
					instruction. The original contents of the	
					destination are lost.	
					A SLO 1 SEGMENT	
					CDA Rd, LABEL (Rd)	
					LDA ARU. LABSSO (F	
					11:1:0:11:1:0	
					THE REPORT OF THE PROPERTY OF	
					Assembler Notation	
					The assembled displacement is a signed	
					two's complement number with a range of +32,767 to -32,768. The label LAB is	
					used by the assembler to generate the	
					displacement relative to the updated PC.	
					A LAB which results in a displacement outside the allowable range produces an	
					assembler error.	
					LOA Rd, Rsf (D)	
					NS.S 0.011.1.0.1.0.1	
					LOA Rd, Hs1 (Rx) US, S (0)1111191101	
					X (See note)	
ags						
	V DA H	Flags are	not affec	ted.		
	- - -					
= Unaffected						
= Set						
ClearedConditional – s						

LDRB

LOAD byte register into memory

LDB dst, Rbs LDRB LAB, Rbs LDB

LDRB

	Version	Mnemonic and Form Clocks	Operation ones M notes M about
			dst<0:7>←Rbs<0:7>
			1 NS. S (1.0.11.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
			M NS S CORERAND
		LDB Rd↑, Rbs	teR John BOJ
IR	NS	0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 Rd Rbs 8	1 10,010,011,010 BM F
		LDB RRd↑, Rbs	LDB Rbd, RRs1
IR	S	0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 RRd Rbs 8	Description
		LDB LABEL, Rbs	The byte contents of the source register are loaded into the byte destination. The
DA	NS	0 1 1 0 1 1 1 0 0 0 0 0 Rbs ADDRESS	source operand is always a general-
		entrustad a breveno	purpose byte register designated by the Rbs field of the instruction. The destina-
		LDB LABSSO, Rbs 0 1 1 0 1 1 1 1 0 0	tion is determined by the applicable
DA	SSO	0 SEGMENT OFFSET	addressing mode. The contents of the source are unaltered and the original
		LDB LABEL, Rbs	contents of the destination are lost.
langi		0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 Rbs	In the IR mode, R0 (or RR0) can be
DA	SLO	1 SEGMENT 14	designated as the general-purpose
		membas etti sepotu	Note: In the BA and BX addressing
		LDB LABEL (Rx), Rbs 0 1 1 0 1 1 1 0 Rx ≠ 0 Rbs	modes the segmented version requires
X	NS	0 1 1 0 1 1 1 0 HX ≠ 0 HDS 12	the designation of a register pair, RRd ≠ 0, as destination base address
		LDB LABSSO (Rx), Rbs	register.
.,		0,1,1,0,1,1,1,0 Rd ≠ 0 Rbs	LUB ADD, LABSSO (EX)
X	SSO	0 SEGMENT OFFSET 12	Assembler Notation
		LDB LABEL (Rx), Rbs	In the RA addressing mode the
X	bis is at mom	0 1 1 1 0 1 1 1 1 0 Rd ≠ 0 Rbs	assembled displacement is a signed two's
^	SLO	1 SEGMENT 15	complement number with a range of +32,767 to -32,768.
		In the RA and rel	In the RA addressing mode the label LAB
		LDRB LAB, Rbs 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 Rbs	is used by the assembler to generate the
RA	NS, S	DISPLACEMENT 14	displacement relative to the updated PC. In the BA addressing mode the value D is
		LDB Rd↑ (D), Rbs	an unsigned integer which is assembled
ВА	NS, S	0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 Rd ≠ 0 Rbs	into the binary displacement.
DA-epsiq	(See note)	DISPLACEMENT 14	A LAB or D which results in a displace- ment outside the allowable range
	tone tek	LDB Rd↑ (Rx), Rbs	produces an assembler error.
	NS, S (See note)	0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 Rd ≠ 0 Rbs Rx 14	XX (See note) 3X
вх			

LDRB

- = Unaffected

1 = Set

0 = Cleared

* = Conditional – see description

LOAD byte into register

LDB Rbd, src LDRB Rbd, LAB LDB

LDRB

Mode	Version	Mnemonic and Form	Clocks	Operation operation and a second
_		LDB Rbd, Rbs		Rbd<0:7>←src<0:7>
R	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 Rbs Rbd	3	
		LDB Rbd, IMb		
18.4		0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 Rbd	_	
IM	NS, S	7 OPERAND 0 7 OPERAND 0	7	
		LDB Rbd, Rs↑		LDB Rut, Rb
IR	NS	$0_10_11_10_10_10_10_10$ Rs $\neq 0$ Rbd	7	NE GOLD TILL
		LDB Rbd, RRs↑		LDS-RRdt, Los
R	S	$0_10_11_10_10_10_10$ RRs $\neq 0$ Rbd	697	Aller Control of the P
		LDB Rbd, LABEL		Description
8111.11		0 ₁ 1 ₁ 1 ₁ 0 ₁		Felal of a Call of the section of the section
DA	NS	ADDRESS	9839	the destination byte register. The source
		I DD DL I ADOCO		operand is determined by the applicable
		LDB Rbd, LABSSO		addressing mode and the destination is
DA	SSO	0 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	70 10	always a general-purpose byte register designated by the Rbd field of the
		rainsh anti la espojena		instruction. The contents of the source
		LDB Rbd, LABEL		operand are unaltered, and the original
DA	SLO	0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 Rbd	12	contents of the destination are lost.
D/Y GEO	SLO	PER US A PRODUCT OFFSET	118-	Note: In the BA and BX addressing
		Note: In the BA and I		modes the segmented version requires
		LDB Rbd, LABEL (Rx)		the designation of a register pair, RRs ≠ 0, as source base address
X	NS	0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 Rx ≠ 0 Rbd ADDRESS	10	register.
		asialore		
		LDB Rbd, LABSSO (Rx)		LDB LABSSO (Rx), R
X	SSO	0 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 Rx ≠ 0 Rbd 0 SEGMENT OFFSET	10	C SSO OT SECURENT
		U SEGMENT OFFSET		Assembler Notation
		LDB Rbd, LABEL (Rx)		In the RA addressing mode the
X		0 1 1 1 0 0 0 0 0 0 Rx \neq 0 Rbd	0 + 6F - 1	assembled displacement is a signed
A	SLO	1 SEGMENT OFFSET	13	two's complement number with a range of +32,767 to -32,768.
				In the RA addressing mode the label LAB
		LDRB Rbd, LAB		is used by the assembler to generate the
RA	NS, S		14	displacement relative to the updated PC.
a train	SV SITE SIDORI	DISPLACEMENT	TVETWS U	In the BA addressing mode the value D is
		LDB Rbd, Rs↑ (D)		an unsigned integer which is assembled
BA	NS, S	$0_10_11_11_10_10_10_10$ Rs $\neq 0$ Rbd	0 = b8 [1	into the binary displacement.
DA	(See note)	DISPLACEMENT	ти14.	A LAB or D which results in a displace-
		LDB Rbd, Rs↑ (Rx)		ment outside the allowable range produces an assembler error.
DV	NS, S	0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 Rs ≠ 0 Rbd		NS18 0.1.4.1.0.0.1.1
BX	(See note)) Rx	14	XA (See note) X8
Flags				

LOAD IMMEDIATE byte into memory

LDB

LDB dst, IMb

Mode	Version	Mnemonic and Form LDB Rd↑, IMb	Clocks	Operation dst<0:7>←IMb<0:7>
IR	NS	0 0 0 0 1 0 1 1 1 1 0 0 Rd 0 1 1 0 1 1 7 OPERAND 0 7 OPERAND 0	11	NS, S [1,1,0,0] Sho
IR	S	LDB RRd↑, IMb 0_10_10_10_11_1_0_0 RRd 0_11_0_1 7 OPERAND 0 7 OPERAND 0	11	
DA	NS	LDB LABEL, IMb 0	14	
DA	SSO	LDB LABSSO, IMb	15	Description The immediate byte operand following the instruction in memory is loaded into the destination. The destination is determined by the applicable addressing
DA	SLO	LDB LABEL, IMb	17	mode. The original contents of the destination are lost. In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.
X	NS	LDB LABEL (Rx), IMb 0	15	
X	SSO	$ \begin{array}{c cccc} LDB \ LABSSO \ (Rx), \ IMb \\ \hline 0_1 1_1 0_1 0_1 1_1 1_1 0_1 0 & Rd \neq 0 & 0_1 1_1 0_1 1 \\ 0 & SEGMENT & OFFSET \\ \hline 7 & OPERAND & 0 & 7 & OPERAND & 0 \\ \hline \end{array} $	15	
X	SLO	LDB LABEL (Rx), IMb 0	18	

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

LOAD IMMEDIATE byte into a register

LDB

LDB Rbd, IMb

Mode	Version	Mnemonic an				Clocks	Operation Rbd<0:7>←IMb<0:7>		
7	NS, S	1,1,0,0	Rbd	1	IMb	5	0.0.1.3.0.0.0.0.0		
							T OPERAND 0		
							1.08 PRot. Stb		
							OF JUNE OF		
							IND LABEL IND		
							0.0:1:1:0:0:1:0:		
							graner (0)		
							Description The immediate byte value	ue in the	
							instruction field, IMb, is destination. The destination general-purpose byte re	loaded into	ys a
							by the Rbd field of the in	struction.	
							THE SEEMENT		
							OPERAND OF		
							UDB LAB 3. 030, IMD		
							DIARESO T		
							MI (XR) OSESAJ SOJ		
							0.1.1.0.0.1.0		
							[7] BEGMENT		
							LOB LAB GL (Rx), IMb		
							10 110 011 10 011		
							T TOPERANO		
Flags	7 0 0	V DA 11		-					
C .	Z S P/	V DA H		Flags	are not affected	J. S. LIGHT			
- = Un	affected								
1 = Se									

LDCTL

LOAD control register from a register

LDCTL

LDCTL CR, Rs

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form LDCTL CR, Rs	Clocks	Operation
R	NS, S	0,1,1,1,1,1,0,1 Rs 1 CR	7	R was constitution
	110, 0			

Description

The CPU control register specified in the CR field of the instruction is loaded from the general-purpose source word register specified by the Rs field of the instruction. The original contents of the control register are lost.

The CR field decodes are shown below:

CR Field	Destination	CR Notation
000	_ sarueā	_ bleld
0 0 1		- 000
0 1 0	FCW	FCW
0 1 1 0 3	Refresh register (bits 1 through 15)	REFRESH
1 0 0	NPSAP segment	PSAPSEG
1 0 1	NPSAP upper offset	PSAPOFF
1 1 0	R14 (normal stack pointer segment)	NSPSEG
1016184	R15 (normal stack pointer offset)	NSPOFF

(Note that the LDCTL instruction is not used to load to the system stack pointer.)

Assembler Notation

The assembler determines the CR field by the reserved word specified according to the table above.

Flags

C Z S P/V DA H

Flags are affected only if the FCW is selected as the destination.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

LDCTL

LOAD control register into a register

LDCTL

LDCTL Rd, CR

This is a SYSTEM instruction.

A NS. S 10:1111111101

Description

The contents of the CPU control register specified in the CR field of the instruction are loaded into the general-purpose destination word register specified by the Rd field of the instruction. The original contents of the destination are lost. Where a control register word of less than 16 bits is loaded into the destination register, zeros are loaded into the unused bit positions.

The CR field decodes are shown below:

Field	Field Source	
Tiold	000100	Notation
0 0 0		- 100
0 0 1	FCW =	- 0 1 0
0 1 0	FCW I riguordi il alidi vetalgen dashleR	FCW
0 1 1	Refresh register (bits 1 through 8)	REFRESH
100	NPSAP segment	PSAPSEG
101	NPSAP upper offset and lamon AFF	PSAPOFF
1 1 0	R14 (normal stack pointer segment)	NSPSEG
1 1 1	R15 (normal stack pointer offset)	NSPOFF

(Note that the LDCTL instruction is not used to load from the system stack pointer.)

Assembler Notation

The assembler determines the CR field by the reserved word specified according to the table above.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

LDCTLB

LOAD flag byte from a register

LDCTLB

LDCTLB FLAGS, Rbs

Mode	Version	Mnemonic and Form	Clocks	Operation
		LDCTLB FLAGS, Rbs		FCW<0:7>←Rbs<0:7>
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 Rbs 1 ₁ 0 ₁ 0 ₁ 1	bd7	Committee of the sister

Description

The flag byte of the FCW is loaded from a general-purpose byte source register designated by the Rbs field of the instruction. The previous contents of the flag register are lost.

Assembler Notation

The assembler notation for the flag byte of the FCW is the reserved word: FLAGS.

Flags

C Z S P/V DA H
* * * * * *

Flags are affected as described above.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

LDCTLB

LOAD flag byte into a register

LDCTLB Rbd, FLAGS

LDCTLB

Mode R	Version NS, S		Rbd	0101011	Clocks	Operation Rbd<0:7>←FCW<0:7>
						Description The flag byte of the FCW is loaded into the general-purpose byte destination register designated by the Rbd field of the instruction. The previous contents of the destination register are lost.
						Assembler Notation The assembler notation for the flag byte of the FCW is the reserved word: FLAGS.
-	Z S P/	V DA H	Flags a	re not affecte	d. 20817	Plags C Z S PN DA H * * * * * * *

LDD

LOAD memory word to memory, autodecrement

LDD

LDD dst, src, Rc

Mode	Version	Mnemonic and Form			Clocks	Operation
		LDD Rd↑, Rs↑, Rc				dst<0:15>←src<0:15>
IR	NC 1	1,0,1,1,1,0,1,1	Rs	1,0,0,1	20	Rs<0:15>←Rs<0:15>-2
IR NS	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd	1,0,0,0	20	Rd<0:15>←Rd<0:15>-2	
		LDD RRd↑, RRs↑, Rc				Rc<0:15>←Rc<0:15>−1
IR	C	1,0,1,1,1,0,1,1	RRs	1,0,0,1	20	11.0.1.1.1.0.1
n	S	0,0,0,0 Rc	RRd	1,0,0,0	20	0.0.0.0 Bc

Description

The source word operand is loaded into the word destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by two.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H

H s at aR philipse P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

LDDB

LOAD memory byte to memory, autodecrement

LDDB

LDDB dst. src. Rc

Mode	Version	Mnemonic and Form			Clocks	Operation dst<0:7> ←src<0:7>	
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rd	1,0,0,1	20	Rs<0:15>←Rs<0:15>−1 Rd<0:15>←Rd<0:15>−1	
IR	S	LDDB RRd↑, RRs↑, Rc 1 0 1 1 1 0 1 0 0 0 0 0 Rc	RRs RRd	1,0,0,1	20	Rc<0:15>←Rc<0:15>-1	

Description

The source byte operand is loaded into the byte destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by one.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H

P/V DA H P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

LOAD memory word to memory, autodecrement and repeat

LDDR

LDDR dst. src. Rc

Mode	Version	Mnemonic and Form LDDR Rd↑, Rs↑, Rc	Clecks		Clocks
IR	NS	1,0,1,1,1,0,1,1 0,0,0,0 Rc	Rs Rd	1 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0	11 + 9n*
IR	S	LDDR RRd↑, RRs↑, Rc 1,0,1,1,1,1,0,1,1 0,0,0,0 Rc	RRs RRd	1,0,0,1	11 + 9n*

*n is the number of iterations.

Operation

dst<0:15> ←src<0:15>
Rs<0:15> ←Rs<0:15> −2
Rd<0:15> ←Rd<0:15> −2
Rc<0:15> ←Rd<0:15> −1
Repeat until termination.

Description

The source word operand is loaded into the word destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by two and the operation will repeat until termination.

Termination occurs when the contents of Rc are zero. This instruction is interruptible.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H
- - - 1 - -

P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

LDDRB

LOAD memory byte to memory, autodecrement and repeat

LDDRB

LDDRB dst, src, Rc

Mode	Version	Mnemonic and Form			Clocks	Operation something age
IR	NS	LDDRB Rd↑, Rs↑, Rc 1,0,1,1,1,1,0,1,0 0,0,0,0 Rc	Rs Rd	1 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0	11 + 9n*	dst<0:7>←src<0:7> Rs<0:15>←Rs<0:15>-1 Rd<0:15>←Rd<0:15>-1 Rc<0:15>←Rc<0:15>-1
IR	S	LDDRB RRd↑, RRs↑, Rc	RRs RRd	1,0,0,1	11 + 9n*	Repeat until termination.

^{*}n is the number of iterations.

Description

In the number of tarations

The source byte operand is loaded into the byte destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by one and the operation will repeat until termination.

Termination occurs when the contents of Rc are zero. This instruction is interruptible.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H

P/V: Set to 1.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

LDI

LOAD memory word to memory, autoincrement

LDI

LDI dst, src, Rc

Mode	Version	Mnemonic and Form			Clocks
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rd	0,0,0,1	20
IR	S	LDI RRd↑, RRs↑, Rc 1 0 1 1 1 0 1 1 0 0 0 0 Rc	RRs RRd	0,0,0,1	20

Operation

dst<0:15>←src<0:15> Rs<0:15>←Rs<0:15>+2 Rd<0:15>←Rd<0:15>+2 Rc<0:15>←Rd<0:15>-1

Description

The source word operand is loaded into the word destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both incremented by two.

R0 can be designated as the general-purpose source or destination register.

Flags

C Z S P/V DA H

P/V DA H P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

LDIB

LOAD memory byte to memory, autoincrement

LDIB

LDIB dst. src. Rc

Mode	Version	Mnemonic and Form	Clocks		Clocks	Operation dst<0:7>←src<0:7>
IR	NS SH	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rd	0,0,0,1	20	Rs<0:15>←Rs<0:15>+ Rd<0:15>←Rd<0:15>- Rc<0:15>←Rc<0:15>-
IR	S	LDIB RRdî, RRsî, Rc 1 0 1 1 1 0 1 0 0 0 0 0 Rc	RRs RRd	0,0,0,1	20	10,011,110,113 0,010,0

Description

The source byte operand is loaded into the byte destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both incremented by one.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H

Has a of page P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- = Set
- 0 = Cleared
- = Conditional see description

LOAD memory word to memory, autoincrement and repeat

LDIR

LDIR dst, src, Rc

Mode	Version	Mnemonic and Form LDIR Rd↑, Rs↑, Rc			Clocks
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rd	$0_10_10_11 \\ 0_10_10_10$	11 + 9n*
IR	S	LDIR RRd↑, RRs↑, Rc 1,0,1,1,1,1,0,1,1 0,0,0,0 Rc	RRs RRd	0,0,0,1	11 + 9n*

^{*}n is the number of iterations.

Operation

 $\begin{array}{l} dst < 0:15> \leftarrow src < 0:15> \\ Rs < 0:15> \leftarrow Rs < 0:15> +2 \\ Rd < 0:15> \leftarrow Rd < 0:15> +2 \\ Rc < 0:15> \leftarrow Rd < 0:15> -1 \\ Repeat until termination. \end{array}$

Description

The source word operand is loaded into the word destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both incremented by two and the operation will repeat until termination.

Termination occurs when the contents of Rc are zero. This instruction is interruptible.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H
- - 1 1 - -

P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

LDIRB

LOAD memory byte to memory, autoincrement and repeat

LDIRB

LDIRB dst, src, Rc

		LDIRB Rd↑, Rs↑, Rc			Clocks	Operation dst<0:7>←src<0:7>
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rd	0,0,0,1	11 + 9n*	Rs<0:15>←Rs<0:15>+1 Rd<0:15>←Rd<0:15>+1
IR		LDIRB RRd↑, RRs↑, Rc	RRs	0,0,0,1	11 + 9n*	Rc<0:15>←Rc<0:15>−1 Repeat until termination.

^{*}n is the number of iterations.

Description

The source byte operand is loaded into the byte destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both incremented by one and the operation will repeat until termination.

Termination occurs when the contents of Rc are zero.

This instruction is interruptible.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H

P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

LDK

LOAD IMMEDIATE digit into a register

LDK Rd, IMd

LDK

SUM YOU DOLL

Mode	Version	Mnemonic and Form			Clocks	Operation	riotaneV	
3	NS, S	LDK Rd, IMd	Rd	IM	7 5	dst<0:3>←IMd<0: dst<4:15>←0	3>	
	110, 0	[,]0],],],],]	110					
						sRR_tbR_2Q2		
					. 68	[0,0,0,1,1,1,0,0]		
						LDL RRdt, RRs		
						101111101010		
						DAR LEBALION		
						10 F F F C F O 102	69	AG
						Description		
						The immediate digit		
						struction field, IMd, least significant four		
						The destination is a	general-purpo	ose word
						register designated instruction. Following		
						bits of the destination		
						P		
						LDL LASET (BX) BB		
						01717110110		
						DC LABSSO (Rx), F		
						D SEGMENT		
						LD: LABEL (RX), RR		
						0,1,1,1,1,0,1		
						1 SEGMENT		
						0		
						LDRL LAB, ARs		
						DISPLA		
						UDL Rdt (D), RRs		
						EDI Rdf (Rx), RRs		
						Belg Fall of the Late O.		
						AH SECTION		
Flags								
C Z	S P/	V DA H	Flags a	re not affecte	d. s again			
	ffected							
= Set	ared							

5-101

LDL dst, RRs

LDL

LDRL

LDRL LAB, RRs

LDRL

Mode	Version	Mnemonic and Form	Clocks	Operation dst<0:31>←RRs<0:31>
IR	NS	LDL Rd1, RRs 0 0 0 1 1 1 1 1 0 1 Rd RRs	11	
IR	S	LDL RRd1, RRs 0,0,0,1,1,1,1,0,1 RRd RRs	11	
DA	NS	LDL LABEL, RRs 0 1 0 1 1 1 0 0 0 0	14	Description The long word contents of the source
DA	SSO	LDL LABSSO, RRs 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 RRs 0 SEGMENT OFFSET	15	register are loaded into the long word destination. The source operand is always a general-purpose long word register pair designated by the RRs field of the instruction. The long word destination is
DA	SLO	LDL LABEL, RRS 0 1 1 0 1 1 1 1 0 1 0 0 0 0 RRS 1 SEGMENT OFFSET	17	determined from the applicable addressing mode. The contents of the source are unaffected and the original contents of the destination are lost.
X	NS	LDL LABEL (Rx), RRs 0 1 1 0 1 1 1 1 1 0 1 Rx ≠ 0 RRs ADDRESS	15	In the mode R0 (or RR0) can be designated as the general-purpose destination register.
X	SSO	LDL LABSSO (Rx), RRs 0,1,0,1,1,1,1,0,1 Rx ≠ 0 RRs 0 SEGMENT OFFSET	15	Note: In the BA and BX addressing modes the segmented version requires the designation of a register pair, RRd = 0, as destination base address register.
X	SLO	DL LABEL (Rx), RRs 0 1 1 0 1 1 1 1 1 0 1 Rx ≠ 0 RRs 1 SEGMENT OFFSET	18	Assembler Notation
RA	NS, S	LDRL LAB, RRs 0,0,1,1,0,1,1,1,0,0,0,0 RRs DISPLACEMENT	17	In the RA addressing mode the assembled displacement is a signed two's complement number with a range of +32,767 to -32,768
BA	NS, S	LDL Rd↑ (D), RRs 0,0,1,1,1,0,1,1,1 Rd ≠ 0 RRs	17	In the RA addressing mode the label LAB is used by the assembler to generate the displacement relative to the updated PC.
DV	(See note)	LDL Rd↑ (Rx), RRs 0_1_1_1_1_0_1_1_1 Rd ≠ 0 RRs		In the BA addressing mode the value D is an unsigned integer which is assembled into the binary displacement.
BX	(See note)	[00000000000] — [00000000000000000000000	17	A LAB or D which results in a displace- ment outside the allowable range produces an assembler error.
Flags C Z	S P/V	DA H Flags are not affecte	d. BagaiR	Plags C 2 8 PV DA H
- = Unat 1 = Set 0 = Clea				

LOAD long word into register LDL LDL LDL RRd. src LDRL LDRL LDRL RRd, LAB Mode Version Mnemonic and Form Clocks Operation LDL RRd. RRs RRd<0:31>←src<0:31> R RRd NS.S 1,0,0,1,0,1,0,0 RRs 5 LDL RRd, IMR 0,0,0,1,0,1,0,0,0,0,0 RRd IM NS.S **OPERAND** 16 11 15 **OPERAND** 0 LDL RRd, Rs1 IR NS 0,0,0,1,0,1,0,0 Rs \(\neq 0 \) RRd 11 LDL RRd, RRs1 IR Description S 0,0,0,1,0,1,0,0 RRs \(\neq 0 \) RRd 11 The source operand long word is loaded LDL RRd, LABEL into the destination long word register. 0,1,0,1,0,1,0,0,0,0,0 RRd The source operand is determined by the DA NS 12 **ADDRESS** applicable addressing mode and the destination is always a general-purpose LDL RRd, LABSSO long word register pair designated by the RRd field of the instruction. The contents DA 13 SSO SEGMENT OFFSET of the source operand are unaltered while the original contents of the destination LDL RRd. LABEL are lost. 0,1,0,1,0,1,0,0 0,0,0,0 Note: In the BA and BX addressing DA SLO SEGMENT 15 modes the segmented version requires **OFFSET** the designation of a register pair, RRs ≠ 0, as source base address LDL RRd, LABEL (Rx) register. $0_11_10_11_10_10_10$ Rx $\neq 0$ RRd X NS 13 **ADDRESS Assembler Notation** LDL RRd, LABSSO (Rx) In the RA addressing mode the $0_11_10_11_10_11_10_10$ Rx $\neq 0$ X assembled displacement is a signed two's SSO 13 SEGMENT **OFFSET** complement number with a range of +32,767 to -32,768. LDL RRd, LABEL (Rx) $0_11_10_11_10_11_10_10$ Rx $\neq 0$ In the RA addressing mode the label LAB X SLO SEGMENT is used by the assembler to generate the 16 displacement relative to the updated PC. **OFFSET** In the BA addressing mode the value D is LDRL RRd, LAB an unsigned integer which is assembled 0,0,1,1,0,1,0,1 0,0,0,0 RRd into the binary displacement. RA NS.S 17 DISPLACEMENT A LAB or D which results in a displacement outside the allowable range LDL RRd, Rs↑ (D) produces an assembler error. $0_10_11_11_10_11_10_11$ Rs $\neq 0$ RRd NS. S BA 17 DISPLACEMENT (See note) LDL RRd, Rs1 (Rx) NS. S 0,1,1,1,0,1,0,1 Rs ≠ 0 RRd BX 17 (See note) Flags Flags are not affected. Z S P/V DA H - = Unaffected 0 = Cleared 1 = Set * = Conditional - see description

LDM

LOAD multiple registers into memory

LDM dst, Rs, N

LDM

Mode	Version	Mnemonic and Form	Clocks	Operation
		LDM Rd↑, Rs, N		dst+n<0:15>←Rs+n<0:15>
IR	NS	0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 0 Rd ≠ 0 1 ₁ 0 ₁ 0 ₁ 1 Rs n	11 + 3N	Repeat for n = 0, 1, 2,, 15.
ID.		LDM RRd↑, Rs, N	0 0 0 0 0 0 0 0 0 0	10,1,0,1,0,0,0 10 11 8.84 Mil
IR	S	Rs n	11 + 3N	40 88 88 10 1
		LDM LABEL, Rs, N	O Ken To	NS NS COLOR OF THE
DA	NS	0,1,0,1,1,1,0,0,0,0,0,0,1,0,0,1 Rs n	14 + 3N	
	110	ADDRESS		12FR BREIDL
		LDM LADCCO De N	0 RRs # (Description
		LDM LABSSO, Rs, N 0 1 0 1 1 1 1 0 0 0		A specified number of general-purpose
DA	SSO	Rs	15 + 3N	word registers are loaded into memory. Loading will take place into consecutive
		LDM LABEL, Rs, N		memory locations with ascending addresses. The first register to be saved
		0,1,0,1,1,1,0,0,0,0,0,0,1,0,0,1		is specified in the Rs field of the
	SIO	Rs n	17 + 3N	instruction and registers will be accessed
DA	SLO	1 SEGMENT	17 + 314	in ascending order, with R0 following R15.
		OFFSET	.0.0.010	The number of registers to be saved is specified in the n field of the instruction. A
	esembles XS : an inclared by	LOWEL ADEL (D.) D. M.		zero in this field represents one register,
		$0_11_10_11_11_10_10$ Rx $\neq 0$ $1_10_10_11$		etc. The destination address is
X	NS	- Rs n	15 + 3N	determined by the applicable addressing
		ADDRESS		mode using the Rd or Rx field of the instruction. The first register will be saved
		LDM LABSSO (Rx), Rs, N		at this address. Succeeding registers will
		$0_11_10_11_11_10_10$ Rx $\neq 0$ $1_10_10_11$		be saved at successive memory locations.
X	SSO	Rs n	15 + 3N	The contents of the general-purpose
		0 SEGMENT OFFSET		registers are not altered.
		LDM LABEL (Rx), Rs, N		This instruction is not interruptible.
		$0_11_10_11_11_10_10$ Rx $\neq 0$ $1_10_10_11$		AN USCAL DATE USE
X	SLO	Rs	18 + 3N	Assembler Notation
	to the test of	1 SEGMENT		The assembler notation N is a numeric
		OFFSET		expression which is assembled into the bit
				field n of the instruction. The range of N is 1 to 16, and $n = N - 1$. Specifying an N
				outside of the allowable range produces
				an assembler error.
				(D) tall ball (D)
				ortion tion as an
				AS (See note)

_				
	0	0	0	
	ıa	м	Ð,	

S P/V DA H

Flags are not affected.

- = Unaffected1 = Set

- 0 = Cleared * = Conditional see description

LOAD multiple registers from memory

LDM

LDM Rd, src, N

Mode	Version	Mnemonic and Form	Clocks
IR	NS	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	11 + 3N
IR	S	LDM Rd, RRs↑, N 0 0 0 1 1 1 1 0 0 RRs ≠ 0 0 0 0 1 Rd	11 + 3N
DA	NS	LDM Rd, LABEL, N 0 1 0 1 1 1 1 0 0 0	14 + 3N
DA	SSO	LDM Rd, LABSSO, N 0	15 + 3N
DA	SLO	LDM Rd, LABEL, N 0 1 0 1 1 1 1 0 0 0	17 + 3N
X	NS	LDM Rd, LABEL (Rx), N 0 1 0 1 1 1 1 0 0 Rx ≠ 0 0 0 0 1 Rd	15 + 3N
X	SSO	LDM Rd, LABSSO (Rx), N 0 1 0 1 1 1 0 0 Rx ≠ 0 0 0 0 1 Rd	15 + 3N
X	SLO	LDM Rd, LABEL (Rx), N 0 1 0 1 1 1 1 0 0 Rx ≠ 0 0 0 0 1 Rd	18 + 3N

Operation

Rd+n<0:15> \leftarrow src+n<0:15> Repeat for n = 0, 1, 2, . . ., 15.

Description

A specified number of general-purpose word registers are loaded with words from consecutive memory locations with ascending addresses. The first register to be loaded is specified in the Rd field of the instruction. The registers will be addressed in ascending order for loading, with R0 following R15. The number of registers to be loaded is specified in the 'n' field of the instruction. A zero in this field represents one register, etc. A source operand address is generated according to the applicable addressing mode. The first register will be loaded from this location. Succeeding registers will be loaded from successive memory locations. The memory contents are not

This instruction is not interruptible.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is 1 to 16, and n=N-1. Specifying an N outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

H Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

LDPS

LOAD program status

LDPS src

LDPS

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation		
				0; f; f; f; 0; 0; 0 p8		
				LOM Rd, BRS1, N		
				0,0,1,1,1,0,0		
Б		LDPS Rs↑		LOM RK, L /BEL, N		
IR	NS	$0_10_11_11_11_10_10_11$ Rs $\neq 0$ $0_10_10_10$	12	0-1-1-1-0-1-2		
R	S	LDPS RRs↑	10	18 3		
n	5	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 RRs ≠ 0 0 ₁ 0 ₁ 0 ₁ 0	16			
		LDPS LABEL		Description		
DA	NS	0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 ADDRESS	16	This instruction loads to status from consecutive locations with ascending	e memory	
		LDPS LABSSO		starting address of the	status is	
DA	SSO	0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 1 0 1 0 1	20	determined by the app mode. In AmZ8001 the	e status is f	our
		LDPS LABEL		consecutive words, an status is two words.	d in Am280)02 the
DA	SLO	0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 0	22	The PC segment number this instruction in ne		
		bbs presegr equice		mode.		
X beb	NS	LDPS LABEL (Rx) 0,1,1,1,1,1,0,0,1 Rx ≠ 0 0,0,0,0 ADDRESS	17	bR L L L L L L L L L L L L L L L L L L L		
		LDPS LABSSO (Rx)		LOM BULLABSSO (F		
X	SSO	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	20	69 03000		
		0 SEGMENT OFFSET	20	THEMORE		
		LDPS LABEL (Rx)		LDM Rd. ABEL (Rx		
X	SLO	0,1,1,1,1,0,0,1 Rx ≠ 0 0,0,0,0 1 SEGMENT	23	0,1,1,1,0,1,0		
		OFFSET	20	Bd_ Bd_ 11 SLGMENT		
		al distribution sees and				

Flags

C Z S P/V DA H
* * * * * *

The processor flags are loaded with the contents of the new FCW.

- = Unaffected

1 = Set

0 = Cleared

LDR

LOAD RELATIVE word into register

LDR

LD Rd, LAB

(See also LD - Load word into register)

Mode	Version	Mnemonic and Form	Clocks	Operation Rd<0:15>←src<0:15>
		LDR Rd, LAB		
	NS.S	0,0,1,1,0,0,0,1 0,0,0,0 Rd	0,0,0,0,0	

Description

The source operand word is loaded into the destination word register. The source operand is determined by the RA addressing mode and the destination is a general-purpose word register designated by the Rd field of the instruction. The contents of the source operand are unaltered while the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to -32,768. The label LAB is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

LDR

0 = Cleared

* = Conditional - see description

LOAD RELATIVE word register into memory

LDR

LDR LAB, Rs

(See also LD - Load word register into memory)

Mode Ve	Mnemonic and Form	Clocks Operation on and show about dst<0:15>←Rs<0:15>
RA NS	LDR LAB, Rs 0 0 1 1 1 1 0 0 1 1 1 1 DISPLACE	
		Description The word contents of the source register are loaded into the word destination. The source operand is a general-purpose word register designated by the Rs field of the instruction. The destination is determined by the RA addressing mode. The contents of the source are unaltered, and the original contents of the destination are lost.
		Assembler Notation The assembled displacement is a signed two's complement number with a range of +32,767 to -32,768. The label LAB is used by the assembler to generate he displacement relative to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.
Flags C Z	S P/V DA H	Flags are not affected.

LDRB

LOAD RELATIVE byte register into memory

LDRB

LDRB LAB, Rbs

(See also LDB - Load byte register into memory)

Mode	Version	Mnemonic and Form	Clocks	Operation Operation
		LDRB LAB, Rbs		dst<0:7>←Rbs<0:7>
RA	NIC C	0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbs	0 0140 0	0,0,0,0,t,1,0,0
nA	NS, S	DISPLACEMENT	CEMENT	U98IG 0.04

Description

The byte contents of the source register are loaded into the byte destination. The source operand is a general-purpose byte register designated by the Rbs field of the instruction. The destination is determined by the RA addressing mode. The contents of the source are unaltered, and the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to -32,768. The label LAB is used by the assembler to generate the displacement that is added to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C Z S P/V DA H

Flags are not affected.

- Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

LDRB

0 = Cleared

* = Conditional - see description

LOAD RELATIVE byte into register

LDRB

LDRB Rbd, LAB

(See also LDB - Load byte into register)

Mode	Version	Mnemonic and Form LDRB Rbd, LAB	Clocks	Clocks	Operation Rbd<0:7>←src<0:7>
RA	NS, S	0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 DISPLA	O O O O O O O O O O O O O O O O O O O	14 MED	A Ne. s A A
		Description The byte contents of the byte contents of the byte source operand is any instruction. The destination of the source are unaity of the source are unaited.			Description The source operand byte is loaded into the destination byte register. The source operand is determined by the RA addressing mode and the destination is a general-purpose byte register designated by the Rbd field of the instruction. The contents of the source operand are unaltered while the original contents of the destination are lost.
		Assembler Notefich The assembled displance's complement nur +32,781 to -32,768, used by the assemble displacement that is a P.C. A LAB which real ment outside the allow produces an assemble			Assembler Notation The assembled displacement is a signed two's complement number with a range of +32,767 to -32,768. The label LAB is used by the assembler to generate the displacement that is added to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.
Flags C 2	′ S P/\	/ DA H	Flags are not aff	ected.	c Z S PIV DA H

LDRL

LOAD RELATIVE long word register into memory

LDRL

LDRL LAB, RRs

(See also LDL - LOAD long word register into memory)

Mode	Version	Mnemonic and Form	Clocks	Operation The American	
		LDRL LAB, RRs		dst<0:31>←RRs<0:31>	
RA	NS, S	0 0 1 1 0 1 1 1 0 0	17	NS.S 0,0,1,1,0,1,0,1	

Description

The long word contents of the source register are loaded into the long word destination. The source operand is a general-purpose long word register designated by the RRs field of the instruction. The destination is determined by the RA addressing mode. The contents of the source are unaltered, and the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to -32,768. The label LAB is used by the assembler to generate the displacement that is added to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

LDRL

LOAD RELATIVE long word into register

LDRL

Mode	Version	Mnemonic and Form	Clocks	Operation
		LDRL RRd, LAB		RRd<0:31>←src<0:31>
RA	NS, S	0,0,1,1,0,1,0,1,0,0,0,0,0 RRd DISPLACEMENT	17 (30)	e Operation and a

Description

The source operand long word is loaded into the destination long word register. The source operand is determined by the RA addressing mode and the destination is a general-purpose long word register designated by the RRd field of the instruction. The contents of the source operand are unaltered while the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to -32,768. The label LAB is used by the assembler to generate the displacement that is added to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

MBIT

MULTIMICRO TEST

MBIT

MBIT

		This is a	SYSTEM instruction.			
Mode _	Version NS, S	Mnemonic and Form MBIT 0	Clocks 11101110 7		Version	Mode
				Description The multimicro input	line $\overline{\mu}$ l is tes	ted.

5

Flags

C Z S P/V DA H

S: Set to 1 if $\overline{\mu l}$ is inactive. Reset otherwise.

Z: Undefined.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

MREQ

MULTIMICRO REQUEST

MREQ

MREQ Ro

This is a SYSTEM instruction.

Mode _	Version NS, S	Mnemonic and Form MREQ Rc 0 1 1 1 1 0 1 1 Rc 1 1	Clocks	Operation See description below.
		*n is the number of decrementation (n = 0 if initial state of $\overline{\mu I}$ was LOV		
				Description
				There is an external input called Micro-In $(\overline{\mu l})$ and an output called Micro-Out $(\overline{\mu O})$. The MREQ instruction tests the state of the $\overline{\mu l}$ input. If the $\overline{\mu l}$ input is LOW, the instruction terminates. If the $\overline{\mu l}$ input is HIGH, the $\overline{\mu O}$ output is activated and the general-purpose register designated by the Rc field of the instruction is decremented by one. The state of the $\overline{\mu l}$ line is tested, and the contents of Rc are repeatedly decremented until they reach zero. The instruction then terminates with the $\overline{\mu O}$ line LOW if $\overline{\mu l}$ is LOW, or with the $\overline{\mu O}$ line HIGH if $\overline{\mu l}$ is HIGH.

			0,	
- *	*	-	-	-

0 0 Instruction terminated after initial test of $\overline{\mu}$ I.
1 Instruction terminated due to contents of Rc reaching zero with $\overline{\mu}$ I HIGH.
1 Instruction terminated due to contents of Rc reaching zero with $\overline{\mu}$ I LOW.

- = Unaffected

1 = Set

0 = Cleared

MRES

MULTIMICRO RESET

MRES

MRES

This is a SYSTEM instruction.

 Mode
 Version
 Mnemonic and Form
 Clocks
 Operation

 −
 NS, S
 0 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1

Description

The multimicro out line $\overline{\mu O}$ is reset HIGH.

5

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

MSET

MULTIMICRO SET

MSET

MSET

a da	Vent	Manuals and Fa	011	Operation
ode	Version	Mnemonic and Form	Clocks	Operation
_	NS, S	MSET 0,1,1,1,1,1,0,1,1,0,0,0,0,1,	0,0,0	μO←LOW and a second s
			0.001001	
				Description The multimicro out line $\overline{\mu O}$ is set LOW. Note that this operation performs an unconditional setting of the $\overline{\mu O}$ line, independent of the state of the multimicro in line $\overline{\mu I}$.
ags C Z	S P/\	/ DA H Flags are not	affected.	Page C Z S PV DA H
= Una = Set	ffected			

MULTIPLY register with word

MULT

MULT RRd, src

Mode	Version	Mnemonic and Form Clocks	
R	NS, S	MULT RRd, Rs	RRd<0:31>←RRd<0:15>xsrc<0:15>
11	110, 0		H WS.S [1:0:0:1:1:0:0:1]
		MULT, RRd, IM	MULTIL FIQA, IMR
IM	NS, S	0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 RRd 70	0 0 10 10 11 11 10 10 10 10
		OPERAND 33	
		MULT RRd, Rs↑	ARESO at
IR	NS	$0_1 0_1 0_1 1_1 1_1 0_1 0_1 1 \text{ Rs} \neq 0 \text{ RRd}$ 70	MULTIL RÖd, Rist
		MULT RRd, RRs1	In ws (cjoiotitioiotiti
IR	S	$0_10_10_11_11_01_011$ RRs $\neq 0$ RRd 70	1-56 K/S ITUM
		10V + C9S BOR 10 + 28S	IR S [0,0,0,1,1,0,0,1] F
		MULT RRd, LABEL	Description
DA	NS	0,1,0,1,1,0,0,1,0,0,0,0 RRd ADDRESS 71	The least significant word of a destination
		ADDRESS //	register pair (multiplicand) is multiplied by
		MULT RRd, LABSSO	the contents of a source word operand (multiplier). The result is loaded into the
DA	SSO	0 1 1 0 1 1 1 0 0 1 0 0 0 0 RRd 72	destination, which is a general-purpose
DA	etaiper scoo	0 SEGMENT OFFSET	register pair, designated by the RRd field
		MULT RRd, LABEL	of the instruction. The source operand is
		0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 RRd	determined from the applicable addressing mode. Both the multiplicand
DA	SLO	1 SEGMENT	and multiplier are treated as signed two's
		OFFSET - 895	complement 16-bit integers. The original
		MULT RRd, LABEL (Rx)	contents of the destination are lost. The
y est.	ation are lost	0.1.0.1.1.0.0.1 Ry ≠ 0 RBd	source contents are unaltered.
X	NS	ADDRESS 72	10,0,0,1,1,0,1,0
		MULT DDd LADCCO (Day)	ACDA X
		MULT RRd, LABSSO (Rx) 0	MULTURON LABSED (R
X	SSO	0 SEGMENT OFFSET 72	11,0,0,1,1,0,1,0
		OFFSET	X sso of segment
		MULT RRd, LABEL (Rx)	
X	SLO		MULTIL ROS, LASEL (RA)
^	SLO	1 SEGMENT 75	X SLO 1 SEGMENT SE
		OFFSET	
			"n is the number of bits equal to one in
			significant trair of the dostination opera

Flags

C	Z	S	P/V	DA	H
*	*	*	0	-	-

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

C: Set to 1 if product is less than -215 or greater than/equal to 215. Reset otherwise.

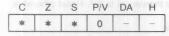
Z: Set to 1 if product is zero. Reset otherwise.S: Set to 1 if product is negative. Reset otherwise.

P/V: Reset.

5-117

		MULTL RQd, RRs	Operation RQd<0:63>←RQd<0:31>xsrc<0:31>
3	NS, S	1 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 RRs RQd 282 + 7n*	Paralitation of the same of th
		MULTL RQd, IMl	
		01010111101010 0101010 RQd	
M	NS, S	31 OPERAND 16 282 + 7n*	
		15 OPERAND 0	
_		MULTL RQd, Rs↑	
R	NS	$0_10_10_11_11_10_10_10$ Rs $\neq 0$ RQd $282 + 7n^*$	
		MULTL RQd, RRs1 05 68A F d > AAA F	B s 0.00 1,1,0,0,1
R	S	$0_10_10_11_11_10_10_10$ RRs $\neq 0$ RQd 282 + 7n*	December
		MULTL RQd, LABEL	Description TAUM
ver ball		0 1 0 1 1 0 0 0 0 0 0 0 0 0	The least significant long word of a destination register quadruple
)A	NS	ADDRESS 283 + 7n*	(multiplicand) is multiplied by the contents
		MULTL RQd, LABSSO	of a source long word operand (multiplier).
Airel b	purinenenaga FIS ad Lud ba	[0.1.0.1.1.0.0.0] POd	The result is loaded into the destination, which is a general-purpose register
DA	SSO	0 SEGMENT OFFSET	quadruple designated by the RQd field of
		MULTL RQd, LABEL	the instruction. The source operand is
		0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 0 ₁ 0 RQd	determined from the applicable addressing mode. Both the multiplicand
DA	SLO	1 SEGMENT 286 + 7n*	and multiplier are treated as signed two's
		OFFSET	complement 32-bit integers. The original
		MULTL RQd, LABEL (Rx)	contents of the destination are lost. The
<	NS	$0_11_10_11_11_10_10_10$ Rx $\neq 0$ RQd $284 + 7n^*$	source contents are unaltered.
`	IVO	ADDRESS	
		MULTL RQd, LABSSO (Rx)	
(SSO	$0_11_10_11_11_10_10_10$ Rx $\neq 0$ RQd $284 + 7n^*$	
`	330	0 SEGMENT OFFSET	
		MULTL RQd, LABEL (Rx)	
		$0_11_10_11_11_10_10_10$ Rx $\neq 0$ RQd	
<	SLO	1 SEGMENT 287 + 7n*	
		OFFSET	
	*n is the n	umber of bits equal to one in the absolute value of the least	
		half of the destination operand.	

Flags



- P/V DA H C: Set to 1 if product is less than -2³¹, or greater than/equal to 2³¹. Reset otherwise.
 - Z: Set to 1 if product is zero. Reset otherwise.
 - S: Set to 1 if product is negative. Reset otherwise.

P/V: Reset.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

NEGATE word

NEG

NEG dst

Mode	Version	Mnemonic and Form NEG Rd			Clocks	Operation dst<0:15>←dst<0:15>	noisteV >+1	
R	NS, S	1,0,0,0,1,1,0,1	Rd	0,0,1,0	7	0,0,1,1,0,0,0,1		
		NEG Rd↑				NEGB Rat		
IR	NS	0,0,0,0,1,1,0,1	Rd	0,0,1,0	12	0,0,0,0,0,0,0,0		
		NEG RRd↑				THE BOSIN		
IR	S	0,0,0,0,1,1,0,1	RRd	0101110	12	6,0,1,1,0,0,6,6		
		NEG LABEL				Description and		
DA	NS	0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ADDR		0 0 0 1 1 1 0	15	The contents of the des		
		NEG LABSSO				complement. The destin		is
DA	SSO	0 1 0 0 1 1 1 0 1 0 1 0 SEGMENT		0 0 0 1 1 1 0 FSET	16		negation is	
DA	SLO	NEG LABEL 0 1 0 0 1 1 0 1 1 1 0 1 1	010101	0 0 0 1 1 1 0	18	destination operand and The original contents of are lost.	the destination	n
	ad nao (097) agogrup-isa	OFFS	ET		13	In the IR mode, R0 (or F designated as the general destination register.		
X	NS	NEG LABEL (Rx) 0 1 0 0 1 1 0 1 ADDR	Rx ≠ 0 ESS	0,0,1,0	16	O O O O O O O O O O O O O O O O O O O		
X	SSO	NEG LABSSO (Rx) 0 1 1 0 0 1 1 1 0 1 0 SEGMENT	Rx ≠ 0	0 ₁ 0 ₁ 1 ₁ 0	0 16 A	NEGRIARSSO (Rx) 0.1.0.0.1.1.0.0 0 SEGMENT		
		NEG LABEL (Rx)	W.			NEGR LARPL (Rx)		
X	SLO	0 1 0 0 1 1 0 1 1 1 0 1 1 SEGMENT	Rx ≠ 0	0,0,1,0	19	0,1,0,0,1,0,0 1) SEGMENT		

Flags

С	Z	S	P/V	DA	Н
*	*	*	*	-	-ceiv

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C Reset on carry from destination. Set to 1 otherwise.
- Z: Set to 1 if the result is zero. Reset otherwise.
 S: Set to 1 if the result is negative. Reset otherwise.
 P/V: Set to 1 if operand value is 8000 (HEX). Reset otherwise.

NEGB

NEGATE byte

NEGB

NEGB dst

Mode R	Version NS, S	Mnemonic and Form NEGB Rbd 1 0 0 0 1 1 0 0 Rbd 0 0 1 0	Clocks 7	Operation dst<0:7>←dst<0:7>+1	
	110, 0				
		NEGB Rd↑		TOR DEM	
R	NS	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 Rd 0 ₁ 0 ₁ 1 ₁ 0 NEGB RRd↑	12	NS (0,0,0,0,0,1,1,0,1)	
R	S	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 RRd 0 ₁ 0 ₁ 1 ₁ 0	12	8 100000000	- 6
		NEGB LABEL 01110101111010 01010 010110		Description The content of the destination between	
DA	NS	ADDRESS	15	The contents of the destination byte operand are replaced by its two's complement. The destination operand is	A
DA	SSO	NEGB LABSSO 0 1 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 0 0 1 1 1 0	16	obtained by using the applicable addressing mode. The negation is	
	nting line d autolog one. The day nati	0 SEGMENT OFFSET NEGB LABEL		achieved by complementing the destination operand and adding one. The original contents of the destination	
DA	SLO	0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0	18	are lost.	
		NEGB LABEL (Rx)		In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.	
X	NS	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	16	011,010,110 EN	
		NEGB LABSSO (Rx)		NEG LAGS 10 (RJ)	
X	SSO	0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0	16	SSO 0.110.0 SECMENT	
		NEGB LABEL (Rx) 0 1 0 0 1 1 1 0 0 Rx ≠ 0 0 0 1 0		NEG LABEL (Rx)	
X	SLO	1 SEGMENT OFFSET	19	TUSINE SECURENT FO	



C	Z	S	P/V	DA	H
*	*	*	*	-	ACT S

- C: Reset on carry from destination. Set to 1 otherwise.
- Z: Set to 1 if the result is zero. Reset otherwise.
 S: Set to 1 if result is negative. Reset otherwise.
 P/V: Set to 1 if the operand value is 80 (HEX). Reset otherwise.
- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

OR

OR word with register

OR

OR Rd, src

Version	Mnemonic and Form OR Rd, Rs	Clocks
NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 Rs Rd	4.0
	OR Rd, IM	
NS, S	0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd OPERAND	7
	OR Rd, Rs↑	
NS	$0_10_10_10_10_11_10_11$ Rs $\neq 0$ Rd	7
S	OR Rd, RRs↑	7
1 (7)	The test of the second	
NS	0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd	9
SSO	0 SEGMENT OFFSET	10
	OR Rd, LABEL	
0.0	0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 Rd	
SLO	1 SEGMENT SEGMENT	12
	OR Rd, LABEL (Rx)	
NS	$0_11_10_10_10_11_10_11$ Rx $\neq 0$ Rd	10
SSO	0 SEGMENT OFFSET	10
	OR Rd, LABEL (Rx)	
	$0_1 1_1 0_1 0_1 0_1 1_1 0_1 1$ Rx $\neq 0$ Rd	
SLO	1 SEGMENT OFFSET	13
	NS, S NS S SSO SLO	Version Minemonic and Form OR Rd, Rs NS, S 1,0,0,0,1,0,1,0,1 Rs Rd OR Rd, IM 0,0,0,0,0,1,0,1,0,1 0,0,0,0,0,0,0 Rd NS 0,0,0,0,0,1,1,0,1 Rs ≠ 0 Rd OR Rd, Rs↑ 0,0,0,0,0,1,1,0,1 RRs ≠ 0 Rd OR Rd, LABEL 0,1,0,0,0,1,1,0,1 RRs ≠ 0 Rd OR Rd, LABEL 0,1,0,0,0,1,1,0,1 0,0,0,0 Rd OR Rd, LABSSO OR Rd, LABESO OFFSET OFFSET OR Rd, LABEL 0,1,0,0,1,1,0,1 0,0,0,0 Rd SLO 1,2,0,0,1,1,0,1 0,0,0,0 Rd OFFSET OR Rd, LABEL OR Rd, LABEL OR Rd, LABEL O1,1,0,0,0,1,1,0,1 0,0,0,0 Rd Rd OFFSET OR Rd, LABEL OR Rd, LABEL OR Rd OR Rd, LABSSO (Rx) OR OR OR OR OR Rd, LABEL (Rx) OR OR <td< td=""></td<>

Description

Operation

Logical OR operation is performed between corresponding bits of the source and destination words. The source operand is obtained using the applicable addressing mode and the destination is always a general-purpose register designated by the Rd field of the instruction. The 16-bit result is loaded into the destination. The source operand is not altered and original destination operand is lost.

Rd<0:15> - src<0:15> V Rd<0:15>

Flags

С	Z	S	P/V	DA	Н
_	*	*	-	-	-

- Z: Set to 1 if result is zero. Reset otherwise.
- S: Set to 1 if result is negative. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- Conditional see description

ORB Rbd, src

Mode	Version	Mnemonic and Form			Clocks
		ORB Rbd, Rbs			
R	NS, S	1,0,0,0,0,1,0,0	Rbs	Rbd	4
		ORB Rbd, IMb			
10.4		0,0,0,0,0,1,0,0	0,0,0,0	Rbd	
IM	NS, S	7 OPERAND 0	7 OPER		819
		ORB Rbd, Rs↑		0,0,0,0	
IR	NS	0,0,0,0,0,1,0,0	Rs ≠ 0	Rbd	olieney to
ID		ORB Rbd, RRs↑	I 55 I		
IR	S	01010101011010	RRs ≠ 0	Rbd	7
		ORB Rbd, LABEL			
DA	NIO	0,1,0,0,0,1,0,0	0,0,0,0	Rbd	•
	NS	ADD	RESS		9
		ORB Rbd, LABSSO			
		0,1,0,0,0,1,0,0	[0.0.0.0]	Rbd	
DA	SSO	0 SEGMENT	OFFS		10
		Vestignation for the Print			
		ORB Rbd, LABEL			
DA	SLO	0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,0,0,0	Rbd	12
		The state of the s	SET	***************************************	12
		Industrial and the street of	OLI		
		ORB Rbd, LABEL (Rx)			
X	NS	0,1,0,0,0,1,0,0		Rbd	10
		ADD	RESS		
		ORB Rbd, LABSSO (R)	()		
X	All que	0,1,0,0,0,1,0,0	Rx ≠ 0	Rbd	10
) Hetalge	SSO	0 SEGMENT	OFF9	SET	10
		ORB Rbd, LABEL (Rx)			
		0,1,0,0,0,1,0,0	Rx ≠ 0	Rbd	
X	SLO	1 SEGMENT	·	**********	13
		OFF	SET		

Description

A logical OR operation is performed between corresponding bits of the source and destination bytes. The source byte is obtained using the applicable addressing mode and the destination byte is always a general-purpose byte register designated by the Rbd field of the instruction. The 8-bit result is loaded into the destination. The source byte is not altered and the original byte in the destination register is lost.

Operationdst<0:7>←src<0:7> V dst<0:7>

С	Z	S	P/V	DA	Н
_	*	*	*	-	-

- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 if parity of result is even. Reset otherwise.

OTDR

- = Unaffected1 = Set0 = Cleared

* = Conditional - see description

OUTPUT word from memory to I/O port, autodecrement and repeat

OTDR

OTDR Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form OTDR Rp, Rs↑, Rc	Clocks	14 0 4 0	Clocks	Operation port dst<0:15> ←src<0:15>
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rp	0101010	11 + 10n*	Rs<0:15>←Rs<0:15>-2 Rc<0:15>←Rc<0:15>-1
IR, PR	S	OTDR Rp, RRs1, Rc	RRs		11 + 10n*	Repeat until termination.
111, 111	3	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rp	0,0,0,0	11 1 1011	
		*n is the number	er of iteration	ons.	Rs ≠ 0	
					PRs ± 0	0,0,1,0,0,0,0,0,0
						Description R 890
					0,0,0,0	A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the
					0,0,0,0 370	instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the
					0,0,0,0 3ET	instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by two. The contents of the general-purpose register
					Rx ≠ 0 RESS	designated by Rc are decremented by one. The instruction is terminated when the result of this decrementation reaches zero.
					0 4 36	This instruction is interruptible.
					990	This instruction uses both indirect register memory addressing and port register port addressing modes.
					Bx ≠ 0 TBP	R0 can be designated as the general- purpose source or port destination register.

OTDRB

OTDRB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form		Clocks	Operation	
IR, PR	NS	OTDRB Rp, Rs1, Rc 0 1 0 1 1 1 1 1 0 1 1 0 0 1 0 1 0 1 0	Rs Rp	1 1 0 1 1 1 0 0 1 0 1 0 1 0 1 1 + 10n*	port dst<0:7>←src<0:7> Rs<0:15>←Rs<0:15>−1 Rc<0:15>←Rc<0:15>−1 Repeat until termination.	
IR, PR	S	OTDRB Rp, RRs1, Rc 0 0 1 1 1 1 1 0 1 1 0 0 1 0 1 0 1 0 Rc	RRs Rp	1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0	0719.79, 9R91, 10 0101111111111111111111111111111111	

^{*}n is the number of iterations.

Description

A data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the generalpurpose register designated by Rs are then decremented by one. The contents of the general-purpose register designated by Rc field are decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose source or port destination register.

Flags

C Z S P/V DA H

P/V: Set to 1.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

OTIR

OUTPUT word to I/O port from memory, autoincrement and repeat

OTIR

OTIR Rp, src, Rc

This is a SYSTEM instruction.

IR, PR S O ₁ O ₁ O ₁ O ₁ O Rc Rp O ₁ O ₁ O ₁ O Discription	Mode Version Mnemonic and Form OTIR Rp, Rs↑, Rc		Clocks	Operation port dst<0:15>←src<0:15>
The second of th	0101010 Rc		11 10 11 + 10n*	Repeat until termination.
Description A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction terminates when the result of this decrementation reaches zero. This instruction is interruptible. This instruction uses both indirect register memory addressing and port register por addressing modes. R0 can be designated as the general-purpose source or port destination register.	IR PR s 0,0,1,1,1,1,0,1,			
A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction terminates when the result of this decrementation reaches zero. This instruction uses both indirect register memory addressing and port register port addressing modes. R0 can be designated as the general-purpose source or port destination register.	*n is the nu	mber of iterations.	riber of Berations.	
contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction terminates when the result of this decrementation reaches zero. This instruction is interruptible. This instruction uses both indirect register memory addressing and port register por addressing modes. R0 can be designated as the general-purpose source or port destination register.				
				contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction terminates when the result of this decrementation reaches zero. This instruction is interruptible. This instruction uses both indirect register memory addressing and port register por addressing modes. R0 can be designated as the general-purpose source or port destination

- = Unaffected1 = Set

0 = Cleared

OTIRB

OUTPUT byte to I/O port from memory, autoincrement and repeat

OTIRB

OTIRB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form OTIRB Rp, Rs↑, Rc		Clocks
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rp	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
IR, PR	S	OTIRB Rp, RRs1, Rc 0 0 1 1 1 0 1 0 0 0 0 0 Rc	RRs Rp	$\begin{array}{ c c c c c c c c c c c c c c c c c c c$

^{*}n is the number of iterations.

Operation

port dst<0:7>←src<0:7>
Rs<0:15>←Rs<0:15>+1
Rc<0:15>←Rc<0:15>-1
Repeat until termination.

Description

A data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one. This instruction terminates when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose source or port destination register.

Flags

C Z S P/V DA H

P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

OUT

OUTPUT word to I/O port from register

OUT

OUT dst, Rs

This is a SYSTEM instruction.

Mode PR	Version NS, S	OUT Rp, Rs 0,0,1,1,1,1,1,1,1	Rp ≠ 0 Rs	Clocks 10	Operation port dst<0:15>←Rs<0:15>
PA	NS, S	OUT PORT, Rs 0_0_1_1_1_1_1_0_1_1_1 PORT ADI	0.0.0.0	12 a8H 0	01010 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
					Description The contents of the general-purpose word source register designated by the Rs field of the instruction are loaded into an output port. The port address destination is determined by the applicable port addressing mode. The source contents are unaltered.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

OUTB

OUTPUT byte to I/O port from register

OUTB

OUTB dst, Rbs

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form OUTB Rp, Rbs	Clocks	Operation port dst<0:7>←Rs<0:7>	
PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 1 ₁ 0 Rp ≠ 0 Rbs	10	tidititition).	
		OUTB PORT, Rbs		0,0,0,0 He	
PA	NS, S	0,0,1,1,1,0,1,0 Rbs 0,1,1,0	12	CUTD Ro. BRs, Rc	
. / .	140, 0	PORT ADDRESS	and I	1,0,1,1,1,0,0	

Description

The contents of the general-purpose byte source register designated by the Rbs field of the instruction are loaded into an output port. The port address destination is determined by the applicable port addressing mode. The source contents are unaltered.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

OUTD

OUTPUT word to I/O port from memory, autodecrement

OUTD

OUTD Rp, src, Rc

This is a SYSTEM instruction.

		No. 11 August 1	e describe			
Mode	Version	OUTD Rp, Rs, Rc	Clocks		Clocks	Operation port dst<0:15>←src<0:15>
R, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rp	1,0,1,0	21	Rs<0:15>←Rs<0:15>-2 Rc<0:15>←Rc<0:15>-1
R, PR	S	OUTD Rp, RRs, Rc 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RRs Rp	1,0,1,0	21	TAOS A
						Description Data word in memory, addressed by the contents of the general-purpose register.
						designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by two. The contents the general-purpose register designated by Rc are decremented by one.
						This instruction uses both indirect register memory addressing and port register por addressing modes. R0 can be designated as the general-purpose source or port destination register.

Flags

C Z S P/V DA H

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

OUTPUT byte to I/O port from memory, autodecrement

OUTDB

OUTDB Rp. src. Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form OUTDB Rp, Rs↑, Rc			Clocks	Operation port dst<0:7>←src<0:7>	
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rp	1,0,1,0	21	Rs<0:15>←Rs<0:15>−1 Rc<0:15>←Rc<0:15>−1	
IR, PR	S	OUTDB Rp, RRs1, Rc 0 0 1 1 1 0 1 0 0 0 0 0 Rc	RRs Rp	1,0,1,0	21	OFT RO. PRST, RO. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10	

Description

Data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose source or port destination register.

Flags

C Z S P/V DA H

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

OUTI

OUTPUT word to I/O port from memory, autoincrement

OUTI

OUTI Rp, src, Rc

This is a SYSTEM instruction.

lode Version	OUTI Rp, Rs↑, Rc		10 0 4 0	Clocks	Operation port dst<0:15> ←src<0:15>
R, PR NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rp	1,0,0,0	21	Rs<0:15>←Rs<0:15>+2 Rc<0:15>←Rc<0:15>-1
R, PR s	OUTI Rp, RRs↑, Rc 0 0 1 1 1 0 1 1 0 0 0 0 Rc	RRs Rp	0,0,1,0	21	0.0708 Rp. RReft, Ro 0.00711 1.00110 0.007010 Rc
					Description A data word in memory, addressed by the
					contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one.
					This instruction uses both indirect register memory addressing and port register port addressing modes. R0 can be designated as the general-purpose source or port destination
					register.

Flags

C Z S P/V DA H

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected

1 = Set

0 = Cleared

OUTPUT byte to I/O port from memory, autoincrement

OUTIB

OUTIB Rp, src, Rc

This is a SYSTEM instruction.

Mode IR, PR	Version NS	Mnemonic and Form OUTIB Rp, Rs↑, Rc 0 1 0 1 1 1 1 1 0 1 1 0 0 1 0 1 0 1 0	Rs Rp	0101110	Clocks 21	Operation port dst<0:7>←src<0:7> Rs<0:15>←Rs<0:15>+1 Rc<0:15>←Rc<0:15>-1
IR, PR s		OUTIB Rp, RRs↑, Rc 0 0 1 1 1 0 1 0 0 0 0 0 Rc	RRs Rp	0,0,1,0	21	POP Rdt. Rat
						POP RESE, Ref
						1.1.0.1.0.0.0
						Description 909
		The word from the me addressed by the gen register designated 5:				Data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the
						instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the
						instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are
						then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one.
						This instruction uses both indirect register memory addressing and port register port addressing modes.
						R0 can be designated as the general- purpose source or port destination register.
						SLO 1 SEGMENT OF

Flags

C Z S P/V DA H

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

POP

POP word

POP

POP dst, Rs↑

Mode	Version	Mnemonic and Form POP Rd, Rs↑	Clocks	Operation and State of State o
R	NS, S	1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 Rs ≠ 0 Rd	8	R, PR as 0,0,0,0 Re
				OUTE Ro. REst. Ro 0,0,1,1,1,1,0,1,1 2,0,0,0,0 Rc
IR	NS	POP Rd↑, Rs↑ 0 0 0 1 0 1 1 1 Rs ≠ 0	12	
IR	S	POP RRd↑, Rs↑ 0 0 0 1 0 1 1 1 Rs ≠ 0 RRd	12	
DA	NS	POP LABEL, Rs↑ 0 1 1 0 1 1 0 1 1 1 1 Rs ≠ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	16	Description The word from the memory location addressed by the general-purpose
DA	SSO	POP LABSSO, Rs↑ 0 1 1 0 1 1 0 1 1 1 1 1 Rs ≠ 0 0 0 0 0 0 0 0 0 SEGMENT OFFSET	16	register designated by Rs, is loaded into the destination. The contents of the register designated by Rs are then automatically incremented by two. Thus,
DA	SLO	POP LABEL, Rs↑ 0 1 1 0 1 1 0 1 1 1 1 Rs ≠ 0 0 0 0 0 0 0 1 SEGMENT :: OFFSET	18	if the general-purpose register designated by Rs is regarded as a stack pointer, then the operation described above can be regarded as a POP. Any general- purpose register except R0 may be
X		POP LABEL (Rx), Rs↑ 0 1 0 1 1 1 Rs ≠ 0 Rx ≠ 0 ADDRESS	16	utilized as a stack pointer. The destination is determined by the applicable addressing mode.
X	SSO	POP LABSSO (Rx), Rs↑ 0 1 1 0 1 1 0 1 1 1 1 Rs ≠ 0 Rx ≠ 0 0 SEGMENT OFFSET	16	
X	SLO	POP LABEL (Rx), Rs↑ 0 1 0 1 1 0 1 1 1 1 1 1 Rs ≠ 0 Rx ≠ 0 1 SEGMENT OFFSET	19	

Flags

C Z S P/V DA H

S P/V DA H Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

POPL dst, Rs↑

Mode	Version	Mnemonic and Form	Clocks	Operation		
		POPL Rd, Rs↑				
R	NS, S	1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 Rs ≠ 0 Rd	12			
			1 Pd =			
		POPL Rdî, Rsî		PUSH Rdf, Rat		
IR	NS	$0_10_10_11_10_11_1 \text{ Rs } \neq 0$ Rd	19			
		POPL RRd1, Rs1				
R	S	$0_10_10_11_10_11_10_11_1 \text{ Rs } \neq 0$ RRd	19	1,0,0,1,0,0,0	8	5
		POPL LABEL, Rsî		Description		
DA	NS	$0_11_10_11_10_11_10_11$ Rs $\neq 0$ $0_10_10_10_1$	22	The long word from	the memory I	location
		ADDRESS	au anu	addressed by the g		
		POPL LABSSO, Rs↑		register designated the destination. The		
-	000	$0_11_10_11_10_11$ Rs $\neq 0$ $0_10_10_10$	w bell his			
JA	SSO		23	register designated		
16 8	550	0 SEGMENT OFFSET	23	automatically incren	nented by four	r. Thus
		0 SEGMENT OFFSET POPL LABEL, Rsî			nented by four se register de	r. Thus signate
e Ad e Nd Bai nett i		0 SEGMENT OFFSET POPL LABEL, Rs↑ 0 1 1 0 1 1 0 1 Rs ≠ 0 0 0 0 0 0 0	i Rd	automatically incren if the general-purpo by Rs is regarded a then the operation of	nented by four se register de is a stack poir described abor	r. Thus esignate nter, ve can
DA		0 SEGMENT OFFSET POPL LABEL, Rsî		automatically incren if the general-purpo by Rs is regarded a then the operation of be regarded as a P	nented by four se register de as a stack poir described abor OP. Any gene	r. Thus esignate nter, ve can eral-
DA	SLO	0 SEGMENT OFFSET POPL LABEL, Rs↑ 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0	i Rd	automatically incren if the general-purpo by Rs is regarded a then the operation of	nented by four se register de as a stack poir described abor OP. Any gene cept R0 may l	r. Thus esignate nter, ve can eral-
DA	SLO	0 SEGMENT OFFSET POPL LABEL, Rs↑ 0 1 1 0 1 1 0 1 1 Rs ≠ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	25	automatically increnif the general-purpo by Rs is regarded a then the operation of be regarded as a purpose register ex utilized as a stack-p destination operand	mented by four se register de as a stack point described about OP. Any gene cept R0 may be pointer. The dis determined	r. Thus esignate nter, ve can eral- be
DA	SLO	0 SEGMENT OFFSET POPL LABEL, Rs↑ 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0	i Rd	automatically incren if the general-purpo by Rs is regarded a then the operation of be regarded as a P purpose register ex- utilized as a stack-p	mented by four se register de as a stack point described about OP. Any gene cept R0 may be pointer. The dis determined	r. Thus esignate nter, ve can eral- be
DA	SLO	0 SEGMENT OFFSET POPL LABEL, Rs↑ 0 1 1 0 1 1 0 1 1 Rs ≠ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	25	automatically increnif the general-purpo by Rs is regarded a then the operation of be regarded as a purpose register ex utilized as a stack-p destination operand	mented by four se register de as a stack point described about OP. Any gene cept R0 may be pointer. The dis determined	r. Thus esignate nter, ve can eral- be
DA X	SLO NS	0 SEGMENT OFFSET POPL LABEL, Rs↑ 0 1 1 0 1 1 0 1 1 0 1 Rs ≠ 0 0 0 0 0 0 0 0 1 SEGMENT OFFSET POPL LABEL (Rx), Rs↑ 0 1 1 0 1 1 0 1 1 0 1 Rs ≠ 0 Rx ≠ 0 ADDRESS POPL LABSSO (Rx), Rs↑ 0 1 1 0 1 1 0 1 1 0 1 Rs ≠ 0 Rx ≠ 0	25	automatically increnif the general-purpo by Rs is regarded atthen the operation of be regarded as a P purpose register ex- utilized as a stack-putilized as a stack-putilized as a stack-putilized applicable addressing	mented by four se register de sa stack poir described abor OP. Any gene cept R0 may loointer. The lis determined mg mode.	r. Thus esignate nter, ve can eral- be
DA X	SLO	0 SEGMENT OFFSET POPL LABEL, Rs↑ 0 1 1 0 1 1 0 1 1 0 1 Rs ≠ 0 0 0 0 0 0 0 1 SEGMENT OFFSET POPL LABEL (Rx), Rs↑ 0 1 1 0 1 1 0 1 1 Rs ≠ 0 Rx ≠ 0 ADDRESS POPL LABSSO (Rx), Rs↑	25	automatically increnif the general-purpoby Rs is regarded at then the operation obe regarded as a purpose register exutilized as a stack-publicable addressing	mented by four se register de as a stack point described about OP. Any gene cept R0 may be pointer. The dis determined	r. Thus esignate nter, ve can eral- be
DA X	SLO NS	0 SEGMENT OFFSET POPL LABEL, Rs↑ 0 1 1 0 1 1 0 1 1 0 1 Rs ≠ 0 0 0 0 0 0 0 0 1 SEGMENT OFFSET POPL LABEL (Rx), Rs↑ 0 1 1 0 1 1 0 1 1 0 1 Rs ≠ 0 Rx ≠ 0 ADDRESS POPL LABSSO (Rx), Rs↑ 0 1 1 0 1 1 0 1 1 0 1 Rs ≠ 0 Rx ≠ 0	25	automatically increnif the general-purpo by Rs is regarded atthen the operation of be regarded as a P purpose register ex- utilized as a stack-putilized as a stack-putilized as a stack-putilized applicable addressing	mented by four se register de sa stack poir described abor OP. Any gene cept R0 may loointer. The lis determined mg mode.	r. Thus esignate nter, ve can eral- be
DA	SLO NS	O SEGMENT OFFSET	25	automatically increnif the general-purpoby Rs is regarded at then the operation obe regarded as a Ppurpose register exutilized as a stack-pdestination operand applicable addressing	mented by four se register de sa stack poir described abor OP. Any gene cept R0 may loointer. The lis determined mg mode.	r. Thus esignate nter, ve can eral- be

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set 0 = Cleared

PUSH

PUSH word

PUSH Rd↑, src

PUSH

Mode	Version	Mnemonic and Form	Clocks			
R	NS, S	PUSH Rd↑, Rs 1 0 0 1 0 0 1 1 Rd ≠ 0 Rs	9			
IM	NS, S	PUSH Rd↑, IM 0	12			
		PUSH Rd↑, Rs↑		teR .16H J9D9		
IR	NS	$0_10_10_11_10_10_11_11$ Rd $\neq 0$ Rs	13	0,1,0,1,0,0,0		
		PUSH Rd↑, RRs↑		teRuteBR 1909		
IR	S	0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 1 Rd ≠ 0 RRs	13	<u>e.h.e.h.e.e.e.e.e.e.e.e.e.e.e.e.e.e.e.e</u>		
		PUSH Rd↑, LABEL		Description		
	NS	0 1 0 1 0 0 1 1 Rd ≠ 0 0 0 0 0 ADDRESS	14	The contents of the reg	nstruction ar	e
		PUSH Rd↑, LABSSO		decremented by two. Toperand is then loaded		
DA	SSO	0 1 1 0 1 1 0 1 0 1 1 1 Rd ≠ 0 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0	14	location addressed by purpose register design	the general- nated in the	AM
		PUSH Rd↑, LABEL		field of the instruction. general-purpose registe		ed by
DA	SLO	0 1 1 0 1 1 0 1 0 1 1 1 Rd ≠ 0 0 0 0 0 0 0 1 SEGMENT	16	Rd is regarded as a state the operation described regarded as a PUSH.	ack pointer, d above car	then be
		PUSH Rd↑, LABEL (Rx)		purpose register excep	t R0 can be	•
X	NS ST	$\begin{array}{ c c c c c c c c c c c c c c c c c c c$	14	utilized as a stack poin operand is determined addressing mode.		
		PUSH Rd↑, LABSSO (Rx)		POPL LARSSO (RX)		
X	SSO		14	0,1,0,1,0,1,0 0 SEGMENT		
		PUSH Rd↑, LABEL (Rx)		FORLLASEL (Rul.)		
		$0_1 1_1 0_1 1_1 0_1 0_1 1_1 1 Rd \neq 0 Rx \neq 0$		0,1,0,1,0,1,0		
X	SLO	1 SEGMENT OFFSET	17	TYBMO33		

Flags

S P/V DA H

Flags are not affected.

- = Unaffected 1 = Set 0 = Cleared * = Conditional see description

PUSH Rd↑, src

	Version	Mnemonic and Form	Clocks	Operation		
R	NS, S	PUSHL Rd↑, Rs 1 0 0 1 0 0 1 Rd ≠ 0 Rs	12	837 86,8		
	140, 0		, ,,,			
IR	NS	PUSHL Rd↑, Rs↑ 0 0 0 1 0 0 1 Rd ≠ 0 Rs	20			
III	NS	$0_10_10_11_10_10_10_11$ Rd $\neq 0$ Rs] 20			
ID		PUSHL Rd↑, RRs↑	1	0.000		
IR	S	$0_10_10_11_10_10_10_11$ Rd $\neq 0$ RRs	20	RST Rdf, B		
		PUSHL Rd↑, LABEL				
DA	NS	$0_11_10_11_10_10_10_11$ Rd $\neq 0$ $0_10_10_10$	20	B. THAT TAR		
DA	NS	ADDRESS	20	[t.t.0.0.0.r.0.0]	2	8
		PUSHL Rd↑ LABSSO		Description		
DA	SSO	$0_11_10_11_10_10_10_11$ Rd $\neq 0$ $0_10_10_10$	21	The contents of the r	egister designa	ated
	000	0 SEGMENT OFFSET	2003	by the Rd field of the		
		PUSHL Rd↑, LABEL		decremented by four word operand is then		
		$0_11_10_11_10_10_10_11$ Rd $\neq 0$ $0_10_10_10$	0.0.0.0	memory location add		Ar
DA	SLO	1 SEGMENT	1 00	The state of the s		
	OLO		23	general-purpose regis		
	020	OFFSET	23	the Rd field of the ins	struction. Thus,	if the
	020		0.0.0.0	the Rd field of the ins general-purpose regis	struction. Thus, ster designated	if the by
X		OFFSET PUSHL Rd↑, LABEL (Rx) $0_11_10_11_0_10_10_11$ Rd $\neq 0$ Rx $\neq 0$	0,0,0,0	the Rd field of the ins	struction. Thus, ster designated stack pointer, t	if the d by hen
X	NS	OFFSET PUSHL Rd↑, LABEL (Rx)	21	the Rd field of the ins general-purpose regis Rd is regarded as a the operation describ regarded as a PUSH	struction. Thus, ster designated stack pointer, the ed above can . Any general-	if the by hen be
×		OFFSET PUSHL Rd↑, LABEL (Rx) $0_11_10_11_0_10_10_11$ Rd $\neq 0$ Rx $\neq 0$	21	the Rd field of the ins general-purpose regis Rd is regarded as a the operation describ regarded as a PUSH purpose register exce	struction. Thus, ster designated stack pointer, the ed above can . Any general- ept R0 can be	if the d by hen be
	NS	OFFSET PUSHL Rd↑, LABEL (Rx) $0 \downarrow 1 \downarrow 0 \downarrow 1 \downarrow 1 \downarrow 0 \downarrow 0 \downarrow 1 \downarrow 1 \downarrow 0 \downarrow 0$	21	the Rd field of the ins general-purpose regis Rd is regarded as a the operation describ regarded as a PUSH purpose register excutilized as a stack po	struction. Thus, ster designated stack pointer, the ed above can . Any general- ept R0 can be sinter. The sour	if the d by hen be
		OFFSET PUSHL Rd↑, LABEL (Rx) $0_11_10_11_10_10_10_11$ Rd $\neq 0$ Rx $\neq 0$ ADDRESS PUSHL Rd↑, LABSSO (Rx)	21	the Rd field of the ins general-purpose regis Rd is regarded as a the operation describ regarded as a PUSH purpose register exce	struction. Thus, ster designated stack pointer, the ed above can . Any general- ept R0 can be sinter. The sour	if the d by hen be
	NS	OFFSET PUSHL Rd↑, LABEL (Rx) $0 \downarrow 1 \downarrow 0 \downarrow 1 \downarrow 1 \downarrow 0 \downarrow 0 \downarrow 1 \downarrow 1 \downarrow 0 \downarrow 0$	21	the Rd field of the ins general-purpose regis Rd is regarded as a sthe operation describ regarded as a PUSH purpose register exceutilized as a stack po operand is determine	struction. Thus, ster designated stack pointer, the ed above can . Any general- ept R0 can be sinter. The sour	if the d by hen be
X	NS SSO	OFFSET PUSHL Rd↑, LABEL (Rx) $ \begin{array}{c cccc} 0_11_10_11_0_10_10_1 & Rd \neq 0 & Rx \neq 0 \\ \hline & ADDRESS \end{array} $ PUSHL Rd↑, LABSSO (Rx) $ \begin{array}{c cccc} 0_11_10_11_10_10_11 & Rd \neq 0 & Rx \neq 0 \\ \hline & 0 & SEGMENT & OFFSET \end{array} $	21	the Rd field of the ins general-purpose regis Rd is regarded as a sthe operation describ regarded as a PUSH purpose register exce utilized as a stack po operand is determine addressing mode.	struction. Thus, ster designated stack pointer, the dabove can and the dabove can be stack pointer. The sound by the application.	if the d by hen be
X	NS	OFFSET PUSHL Rd↑, LABEL (Rx) $0_11_10_11_10_10_10_11$ Rd $\neq 0$ Rx $\neq 0$ ADDRESS PUSHL Rd↑, LABSSO (Rx) $0_11_10_11_10_10_10_11$ Rd $\neq 0$ Rx $\neq 0$ 0 SEGMENT OFFSET PUSHL Rd↑, LABEL (Rx) $0_11_10_11_10_10_10_11$ Rd $\neq 0$ Rx $\neq 0$ 0 SEGMENT	21 21	the Rd field of the ins general-purpose regis Rd is regarded as a st the operation describ regarded as a PUSH purpose register exce utilized as a stack po operand is determine addressing mode.	struction. Thus, ster designated stack pointer, the ed above can . Any general- ept R0 can be sinter. The sour	if the d by hen be
X	NS SSO	OFFSET PUSHL Rd↑, LABEL (Rx) $ \begin{array}{c cccc} 0_11_10_11_0_10_10_1 & Rd \neq 0 & Rx \neq 0 \\ \hline & ADDRESS \end{array} $ PUSHL Rd↑, LABSSO (Rx) $ \begin{array}{c cccc} 0_11_10_11_10_10_10_1 & Rd \neq 0 & Rx \neq 0 \\ \hline & SEGMENT & OFFSET \end{array} $ PUSHL Rd↑, LABEL (Rx) $ \begin{array}{c cccc} 0_11_01_10_10_10_1 & Rd \neq 0 & Rx \neq 0 \\ \hline & O_11_01_10_10_10_1 & Rd \neq 0 & Rx \neq 0 \end{array} $	21 21	the Rd field of the ins general-purpose regis Rd is regarded as a sthe operation describ regarded as a PUSH purpose register exce utilized as a stack po operand is determine addressing mode.	struction. Thus, ster designated stack pointer, the dabove can and the dabove can be stack pointer. The sound by the application.	if the d by hen be
x x	NS SSO	OFFSET PUSHL Rd↑, LABEL (Rx) $0_11_10_11_10_10_10_11$ Rd $\neq 0$ Rx $\neq 0$ ADDRESS PUSHL Rd↑, LABSSO (Rx) $0_11_10_11_10_10_10_11$ Rd $\neq 0$ Rx $\neq 0$ 0 SEGMENT OFFSET PUSHL Rd↑, LABEL (Rx) $0_11_10_11_10_10_10_11$ Rd $\neq 0$ Rx $\neq 0$ 0 SEGMENT	21 21	the Rd field of the ins general-purpose regis Rd is regarded as a set the operation describ regarded as a PUSH purpose register exce utilized as a stack po operand is determine addressing mode.	struction. Thus, ster designated stack pointer, the dabove can and the dabove can be stack pointer. The sound by the application.	if the d by hen be

Flags

C Z S P/V DA H

Flags are not affected.

Unaffected

1 = Set

0 = Cleared

RES

RESET bit in word (static)

RES dst, B

RES

Mode	Version	Mnemonic and Form RST Rd, B	Clocks	Operation word dst <b bit="">←0
R	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 Rd b	4	NS.6 (1:01011;010,011
				PUSHL Rat Rst
				3 NS 0,0,1,0,0,1,0,0,1
				tean the lifeting
IR	NS	RST Rdî, B 0 0 0 1 1 1 0 1 0 1 0 1 1 1 1 Rd ≠ 0 b	11	1,0,0,0,1,0,0,0
II .	INO] ''	PUSHL Rdt. LABEL
IR	S	RST RRd1, B 0,0,1,0,0,0,1,1 RRd ≠ 0 b	0 N DR	1,0,0,0 t,0,1,0
in	5	0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 RRd ≠ 0 b	11:334	
		RST LABEL, B	1	Assembler Notation
	NS	0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 b ADDRESS	13	The assembler notation B is a numerical expression which is assembled into a
		and ad heterograph	H-RU 1	binary value in the b field of the
		RST LABSSO, B 0,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0	1	instruction. The range of B is zero through
DA	SSO	0 SEGMENT OFFSET	14	15, and b = B. Specifying a B outside of the allowable range produces an
		RST LABEL, B	SET	assembler error.
		0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 b	1	PUSHL BO LABEL IN
DA	SLO	1 SEGMENT	16	1.0.0.0.1.0.1.0
		OFFSET	RESS	ADD
		TIOT LADEL (TIX), D	(Hx)	PUSHL Ret. LABSSO
	NS	$0_11_11_10_10_10_11_11$ Rx $\neq 0$ b	149	1(0,0)0(1,0,1,0) 022
		ADDRESS	HRO I	(O SESMENT
		RST LABSSO (Rx), B	(x	PUSHL RAT, LABEL (F
X	SSO	0 1 1 1 1 0 1 0 1 0 1 1 1	14	1 25 0 10 1 10 10 10 10 10 10 10 10 10 10 10
			192	SLO 1 SEGMENT
		RST LABEL (Rx), B 0 1 1 1 1 0 1 0 1 0 1 1 1 Rx ≠ 0 b	1	
X	SLO	1 SEGMENT	17	
		OFFSET		

Flags

S P/V DA H

Flags are not affected.

- = Unaffected1 = Set

0 = Cleared

RES

RESET bit in word (dynamic)

RES

RES Rd, Rs

Mode	Version	Mnemonic and Form RES Rd, Rs			Clocks	Operation Rd bit specified in Rs(0:3)>←	
R	NS, S	0,0,1,0,0,0,1,1 Rd	0,0,0,0	Rs	10	1,7,0,8,0,7,0,11 8.8	
						9,158-58-3E	
						0,0,0,0,0,0,0 BESB-8841,B	
						110000000000000000000000000000000000000	9
						Description	
						The selected bit of the word de reset to zero. The destination w operand is the general-purpose designated by the Rd field of the instruction. The bit of the destin	vord e register ne nation
						register to be reset is determine binary decode of the least signi bits of a general-purpose word designated by the Rs field of the instruction. The remaining 15 b	ificant four register ie
						destination are unaltered.	
						10A (10,0,0,1,1,0)	
						RESIARSO (NA. 10.0.1.1 (N. 1.1.0.1.1 (N. 1.1.1	
						RESELANCE (MY SECONDAL)	
Flags C Z	S P/N	/ DA H	Flage are	not affecte	ad.		
	J - -	/ DA H	riays are	not affects	u.		
- = Unaf	fected						
- Unai	iected						

5-139

Mode	Version	Mnemonic and Form RESB Rbd, B	Clocks	Operation byte dst b bit>←0
3	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 Rbd b	0.0.4	Б. В.
R	NS	RESB Rd \uparrow , B $0_{1}0_{1}1_{1}0_{1}0_{1}0_{1}1_{1}0$ Rd \neq 0 b	11	
		RESB RRd↑, B		
R	S	0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 RRd ≠ 0 b	11	Description
DA	NS	RESB LABEL, B 0 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0	13	The selected bit of the byte destination is reset to zero. The remaining seven bits are unaltered. The destination is
DA	SSO	RESB LABSSO, B 0 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0	14	determined by the applicable addressing mode, while the bit to be reset is determined by the binary value of the b field of the instruction.
16/8	SLO SLO	RESB LABEL, B 0 1 1 1 1 0 0 0 1 1 1 0 0 0 1 0 1 0 b	16	nera of the instruction.
(NS	RESB LABEL (Rx), B 0 1 1 0 0 1 0 Rx ≠ 0 b ADDRESS	14	
<	SSO	RESB LABSSO (Rx), B	14	Assembler Notation The assembler notation B is a numeric
×	SLO	RESB LABEL (Rx), B 0 1 1 1 0 1 0 1 1 1 0	17	expression which is assembled into a binary value in the b field of the instruction. The range of B is zero through 7, and b = B. Specifying a B outside of the allowable range produces an assembler error.
lags	Z S P/	/ DA H Flags are not affected.		
C Z				
- -	- - -			
- -	affected			

RESB

0 = Cleared

* = Conditional - see description

RESET bit in byte (dynamic)

RESB

Mode R	Version	RESB F	nic and Form Rbd, Rs _0_0_0_1_0 Rbd	0101010 Rs	Clocks	Operation Rbd <bit in="" rs(0:2)="" specified="">←0</bit>
						reset to zero. The destination byte operand is the general-purpose register designated by the Rbd field of the instruction. The bit of the destination
						Assembler Motation The assembler notation LIST refers to a list of or OV. Note that PY and OV affect the same fr
Flags						epal ²

RESFLG

RESET FLAGS

RESFLG LIST

RESFLG

Mode	Version	Mnemonic and Form RESFLG LIST	Clocks	Operation 500 CO. T.O. P.O.	
_	NS, S	1,0,0,0,1,1,0,1 C,Z,S,PV 0,0,1,1	9 7 0 0	FCW <c;z;s;p v="">←0 (see description below)</c;z;s;p>	
	110, 0			Ed9	

Description

The CPU flags C, Z, S and P/V are reset or unaltered, according to the bit settings in the instruction field as described in the table below

Instruction Bit	If = 0	If = 1	Assembler Notation
osto a7d	No effect	Reset C flag	CY
60	No effect	Reset Z flag	ZR
5	No effect	Reset S flag	SGN
4	No effect	Reset P/V flag	PV or OV

Assembler Notation

The assembler notation LIST refers to a list of any or all of the following reserved words, separated by commas: CY, ZR, SGN, PY or OV. Note that PY and OV affect the same flag.

Flags

C Z S P/V DA H

* * * * - -

See above.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

RETURN conditional from subroutine RET RET RET CC Mode Version Mnemonic and Form Clocks Ulmemonic and Form RET CC CC True/CC False NS, S 1,0,0,1,1,1,1,0,0,0,0,0 CC 10,13 7,7 Operation Non-segmented Segmented If CC condition met: If CC condition met: PC←(R15<0:15>) PC segment←(RR14<0:22>) R15<0:15>←R15<0:15>+2 R15<0:15>←R15<0:15>+2 PC OFFSET←(RR14<0:22>) R15<0:15> -R15<0:15>+2 Otherwise: Otherwise: PC←PC+2 PC OFFSET←PC OFFSET+2 Note: In the system mode R15' and RR14' are used instead of R15 and RR14, respectively.

Description

This instruction conditionally returns the CPU to the calling program. During a subroutine call the return address was automatically stacked. This return address is popped from the stack into the PC to effect the return. If the flags do not satisfy the conditions specified by the CC field, the PC is not loaded with the return address but merely updated to the following instruction. The stack pointer remains unaltered from its original value if there is no return.

Assembler Notation

Specifying condition CC is optional. If none is specified, the CC field of the instruction is set to hex eight.

Flags

C Z S P/V DA H

P/V DA H Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

lode	Version	Mnemonic and Form RL Rd, B	Clocks	Operation Operation
	NS, S	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 1 Rd 0	101010	NS. S 1 . 0 . 0 . 1 1 . 1 . 1 . 1 . 1 . 1
			6 (one place) 7 (two places)	C 15 0
				Description The contents of the general-purpose word register designated by the Rd field of the instructions are rotated left. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.
			us 81 Fi to beeteni bebu	Assembler Notation The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B - 1. Specifying a B
				outside of the allowable range produces an assembler error.
			of loaded with the reti	
			getted, the CC field of	
ags	Z S P/\	/ DA H C: Loaded from	m last bit rotated out o	f dectination register
	* * *	Z: Set to 1 if r	esult is zero. Reset of esult is negative. Rese	herwise.

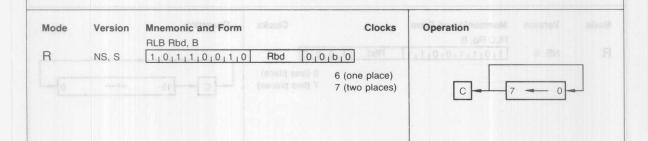
- = Unaffected1 = Set0 = Cleared

RLB

ROTATE byte left

RLB Rbd, B

RLB



Description

The contents of the general-purpose byte register designated by the Rbd field of the instruction are rotated left. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B - 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H
* * * * - -

- P/V DA H C: Loaded from last bit rotated out of destination register.
 - Z: Set to 1 if result is zero. Reset otherwise.
 - S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 if sign of register changed during rotation. Reset otherwise.

- Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

RLC

ROTATE word left through carry

RLC Rd, B

RLC

Mode Version Mnemonic and Form Clocks RLC Rd, B R NS. S Rbd 1,0,1,1,0,0,1,1 1,0,b,0

6 (one place) 7 (two places) Operation

0

Description

The contents of the destination word register designated by the Rd field of the instruction are rotated one or two places left. The last bit shifted out of the destination word is loaded into the carry flag, while the previous contents of the carry flag are shifted into the least significant bit of the destination word. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B - 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

DA * * *

- C: Loaded from the last bit rotated out of destination register.
- Set to 1 if result is zero. Reset otherwise.
- Set to 1 if result is negative. Reset otherwise.
- P/V: Set to 1 if sign of destination contents changed during rotation. Reset otherwise.

= Unaffected

1 = Set

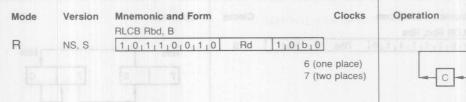
0 = Cleared

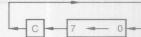
RLCB

ROTATE byte left through carry

RLCB Rbd, B

RLCB





Description

The contents of the destination byte register designated by the Rbd field of the instruction are rotated one or two places left. The last bit shifted out of the destination byte is loaded into the carry flag, while the previous contents of the carry flag are rotated into the least significant bit of the destination byte. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b=B-1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

P/V DA H C: Loaded from the last bit rotated out of destination.

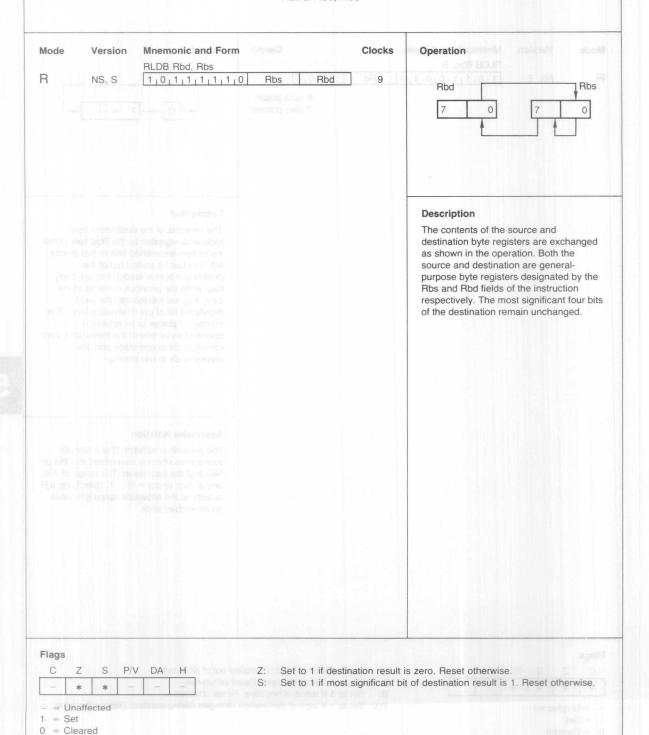
Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 if sign of destination changed during rotation. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

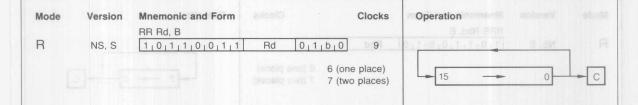
RLDB Rbd, Rbs



ROTATE word right

RR Rd, B

RR



Description

The contents of the general-purpose word register designated by the Rd field of the instructions are rotated right. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b=B-1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

P/V DA H Loaded from last bit rotated out of destination register.

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

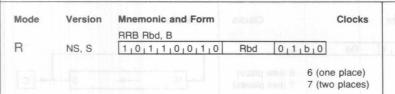
P/V: Set to 1 if sign of destination register changed during rotation. Reset otherwise.

RRB

ROTATE byte right

RRB Rbd. B

RRB



Operation



Description

The contents of the general-purpose byte register designated by the Rbd field of the instructions are rotated right. The number of places to be rotated is specified by bir one of the instructions; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B - 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

* * * * - -

- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

DA H C: Loaded from least significant bit rotated out of destination register.

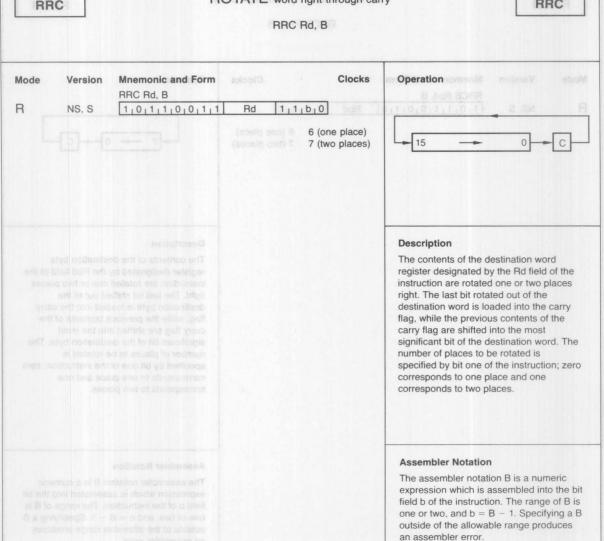
Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 if sign of destination register changed during rotation. Reset otherwise.

ROTATE word right through carry

RRC



an assembler error.

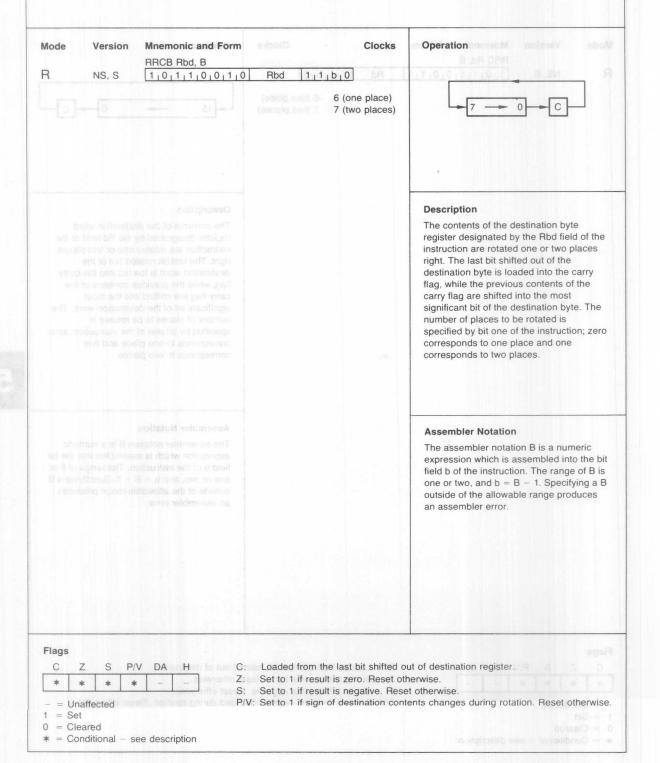
Flags

* *

- P/V DA H C: Loaded from the last bit rotated out of destination.
 - Z: Set to 1 if result is zero. Reset otherwise.
 - S: Set to 1 if result is negative. Reset otherwise.
 - P/V: Set to 1 if sign of register changed during rotation. Reset otherwise.

- - Unaffected

- 1 = Set
- 0 = Cleared
- * = Conditional see description

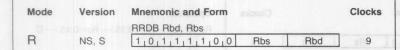


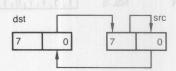
RRDB

ROTATE DIGIT RIGHT, byte

RRDB

RRDB Rbd, Rbs





Description

Operation

The contents of the source and destination byte register are exchanged as shown in the operation. Both the source and destination are general-purpose byte registers designated by the Rbs and Rbd fields of the instruction, respectively. The most significant four bits of the destination remain unchanged.

Flags

C Z S P/V DA H

- Z: Set to 1 if destination result is zero. Reset otherwise.
 - S: Set to 1 if most significant bit of destination result is 1. Reset otherwise.

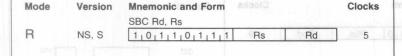
- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

SBC

SUBTRACT word with carry

SBC Rd, Rs

SBC



Operation 1997

Rd<0:15>←Rd<0:15>-Rs<0:15>-C

Description

The source operand word is subtracted from the destination operand word, along with carry, to obtain the result. The subtraction is achieved by adding the two's complement of the source operand to the destination operand.

Both the source and destination are general-purpose word registers designated by the Rs and Rd fields of the instruction, respectively. The 16-bit result is loaded into the destination register, whose original contents are lost. The contents of the source are not affected.

Flags

C Z S P/V DA H

* * * * - -

- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

H of class and C: Reset to 0 on carry from most significant bit of result. Set otherwise.

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 if there is arithmetic overflow. Reset otherwise.

SBCB

SUBTRACT byte with carry

SBCB Rbd, Rbs

SBCB

Mode Versi	on Mnemonic and Form SBCB Rbd, Rbs	Clocks	Operation Rbd<0:7>←Rbd<0:7>-Rbs<0:7>-C
R NS, S		5	парадалица (о) — а жи
		150 - 2 150 - 2 150 - 2 12)	Description The source operand byte is subtracted from the destination operand byte, along with carry, to obtain the result. The subtraction is achieved by adding the two's complement of the source operand to the destination operand. Both the source and destination are general-purpose byte registers designated by the Rbs and Rbd fields of the instruction, respectively. The 8-bit result is loaded into the destination register whose original contents are lost. The contents of the source are not altered.
		agrate counter of from the NiF dion is user of or which is as	ds the new processor globe using NPSAP, or status stored on the state comprises the processor desired to instruction used. As the standers of contral. The Secretary Sets of the instruction occurred. The Secretary Sets of the instruction occurred.

= Unaffected

*

1 = Set

0 = Cleared

* = Conditional - see description

P/V DA H

- C: Reset to 0 on carry from most significant bit of result. Set otherwise.
- Z: Set to 1 if result is zero. Reset otherwise.S: Set to 1 if result is negative. Reset otherwise.
- P/V: Set to 1 if there is arithmetic overflow. Reset otherwise.

DA: Set to 1 always.

H: Reset to 0 if there is a carry from most significant bit of the lower 4 bits of the result. Set otherwise.

SC

SYSTEM CALL

SC N

SC

Mode	Version	Mnemonic and Form		Clocks	
		SC N			
	NS, S	0,1,1,1,1,1,1,1	a n bdR	33, 39	

Operation

Non-Segmented

R15′<0:15>←R15′<0:15>−2 (R15′<0:15>)←PC<0:15>+2

> R15'<0:15> \leftarrow R15'<0:15>-2 (R15'<0:15>) \leftarrow FCW R15'<0:15> \leftarrow R15'<0:15>-2 (R15'<0:15>) \leftarrow Identifier FCW \leftarrow (NPSAP<0:15>+12) PC \leftarrow (NPSAP<0:15>+14)

Segmented

 $\begin{array}{l} \text{R15'}{<}0\text{:15}{>}{\leftarrow}\text{R15'}{<}0\text{:15}{>}{-2} \\ (\text{RR14'}{<}0\text{:22}{>}){\leftarrow}\text{PC OFFSET}{+2} \\ \text{R15'}{<}0\text{:15}{>}{\leftarrow}\text{R15'}{<}0\text{:15}{>}{-2} \\ (\text{RR14'}{<}0\text{:22}{>}){\leftarrow}\text{PC SEGMENT} \\ \text{R15'}{<}0\text{:15}{>}{\leftarrow}\text{R15'}{<}0\text{:15}{>}{-2} \\ (\text{RR14'}{<}0\text{:22}{>}){\leftarrow}\text{FCW} \\ \text{R15'}{<}0\text{:15}{>}{-2} \\ (\text{RR14'}{<}0\text{:22}{>}){\leftarrow}\text{Hothifier} \\ \text{FCW} \leftarrow (\text{NPSAP}{<}0\text{:22}{>}{+26}) \\ \text{PC SEGMENT} \leftarrow (\text{NPSAP}{<}0\text{:22}{>}{+28}) \\ \text{PC OFFSET} \leftarrow (\text{NPSAP}{<}0\text{:22}{>}{+30}) \end{array}$

Description

This instruction produces a system call trap. The system call causes the program status to be pushed into the system stack and then loads the new processor status using NPSAP.

The status stored on the stack comprises the program counter return address, and the flag control word (FCW) as well as the system call instruction itself, as the Identifier.

The new program counter and FCW are obtained from the NPSAP and are loaded into the relevant CPU registers to cause the transfer of control. The 8-bit n field of the instruction is user definable, and thus allows up to 256 identifiers.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into a binary value in the n field of the instruction. The range of N is zero to 255, and n=N. Specifying an N outside the allowable range produces an assembler error.

Flags

C Z S P/V DA H * * * * * * * As specified by the new FCW.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

SDA

SHIFT word arithmetic (dynamic)

SDA

SDA Rd. Rs.

Mode	Version	Mnemonic and Form			Clocks
R	NS, S	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 1 Rs	Rd	1,0,1,1	15 + 3n*

*n is the number of places shifted

Operation

Rd<0:15>←Rd<0:15>shifted

Description

The contents of a general-purpose word register designated by the Rd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range -16 to +16.

This operation is identical to the operation SDL apart from the treatment of the most significant bit of the word, bit 15. This bit is unaltered during right shifts, and shifts into the adjacent bit, bit 14. For left shifts, the bit is treated in an identical manner to other bits of the register. Thus a signed operand has the sign preserved during right shifts.

Flags

C	Z	S	P/V	DA	Н
*	*	*	*	-	_

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Loaded from bit 15 shifted out of destination register (left shift) or from bit 0 shifted out of the destination register (right shift).
 - Z: Set to 1 if the result is zero. Reset otherwise.
 - S: Set to 1 if most significant bit of resultant destination register is 1. Reset otherwise.
 - P/V: Set to 1 if sign of destination register is changed during shift. Reset otherwise.

SDAB

SHIFT byte arithmetic (dynamic)

SDAB Rbd. Rs

SDAB

Mode	Version	Mnemonic and Form SDAB Rbd, Rs			Clocks
R	NS, S	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0	Rbd	1,0,1,1	15 + 3n*

*n is the number of places shifted.

Operation

Rbd<0:7>←Rbd<0:7>shifted

Description

The contents of a general-purpose byte register designated by the Rbd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range -8 to +8.

This operation is identical to the operation SDLB apart from the treatment of the most significant bit of the byte, bit seven. This bit is unaltered during right shifts, and shifts into the adjacent bit, bit six. For left shifts, the bit is treated in an identical manner to other bits of the register. Thus a signed operand has the sign preserved during right shifts.

Flags

C	Z	S	P/V	DA	Н
*	*	*	*	- 1	-

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Loaded from bit 7 shifted out of destination register (left shift) or from bit 0 shifted out of the destination register (right shift).
- Z: Set to 1 if the result is zero. Reset otherwise.
- S: Set to 1 if most significant bit of resultant destination register is 1. Reset otherwise.
- P/V: Set to 1 if sign of destination register is changed during shift. Reset otherwise.

SDAL

SHIFT long word arithmetic (dynamic)

SDAL

SDAL RRd, Rs

Mode	Version	Mnemonic and Form	Clocks	Operation
		SDAL RRd, Rs		RRd<0:31>←RRd<0:31>shifted
R	NS, S	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 1 RRd 1 ₁ 1 ₁ 1 ₁ 1 Rs	15 + 3n*	

*n is the number of places shifted.

Description

The contents of a general-purpose long word register designated by the RRd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range -32 to +32.

This operation is identical to the operation SDLL apart from the treatment of the most significant bit of the long word, bit 31. This bit is unaltered during right shifts, and shifts into the adjacent bit, bit 30. For left shifts, the bit is treated to other bits of the register. Thus a signed operand has the sign preserved during right shifts.

Flags

С	Z	S	P/V	DA	H
*	*	*	*	estw	ianito

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Loaded from bit 31 shifted out of destination register (left shift) or from bit 0 shifted out of the destination register (right shift).
- Z: Set to 1 if the result is zero. Reset otherwise.
- S: Set to 1 if most significant bit of resultant destination register is 1. Reset otherwise.
- P/V: Set to 1 if sign of destination register is changed during shift. Reset otherwise.

Mode	Version	Mnemonic and Form SDL Rd, Rs			Clocks
R	NS, S	1,0,1,1,0,0,1,1 Rs	Rd	0,0,1,1	15 + 3n*

*n is the number of places shifted.

Operation

Rd<0:15>←Rd<0:15>shifted

Description

The contents of a general-purpose word register designated by the Rd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range -16 to +16.

Flags

6	C	Z	S	P/V	DA	Н
	*	*	*	*	-	-

= Unaffected

= Set and Mask hids grant bag

= Cleared

= Conditional - see description

C: Loaded from the last bit shifted out of the destination register.

Z: Set to 1 if the result is zero. Reset otherwise.

S: Set to 1 if the result is negative. Reset otherwise.

P/V: Undefined.

SDLB

SHIFT byte logical (dynamic)

SDLB Rbd. Rs

SDLB

Mode	Version	Mnemonic and Form			Clocks
		SDLB Rbd, Rs			
B	NS, S	1,0,1,1,0,0,1,0	Rbd	0,0,1,1	15 + 3n*
11	140, 0	Rs :			15 011

*n is the number of places shifted.

Operation

 $Rbd < 0:7 > \leftarrow Rbd < 0:7 > (shifted)$

Description

The contents of a general-purpose byte register designated by the Rbd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range -8 to +8.

Flags

C Z S P/V DA H

* * * * - -

- = Unaffected

1 = Set

0 = Cleared

★ = Conditional - see description

C: Loaded from the last bit shifted out of the destination register.

Z: Set to 1 if the result is zero. Reset otherwise.

S: Set to 1 if the result is negative. Reset otherwise.

P/V: Undefined.

SDLL

SHIFT long word logical (dynamic)

SDLL RRd. Rs

SDLL

Mode	Version	Mnemonic and Form	Clocks
		SDLL RRd, Rs	
D	110.0	1,0,1,1,0,0,1,1 RR	d 0 ₁ 1 ₁ 1 ₁ 1 15 + 3n*
Н	NS, S	Rs	15 + 3n*

*n is the number of places shifted.

Operation RRd<0:31>←RRd<0:31>(shifted)

Description

The contents of a general-purpose register pair designated by the RRd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range -32 to +32.

Flags

C	Z	S	P/V	DA	Н
*	*	*	*	- Silv	ne Tio

- = Unaffected
- 1 = Set
- 0 = Cleared
- = Conditional see description

- Loaded from the last bit shifted out of destination register pair.
- Set to 1 if the result is zero. Reset otherwise.
- Set to 1 if the result is negative. Reset otherwise.

P/V: Undefined.

SET

SET bit in word (static)

SET dst, B

SET

Mode	Version	Mnemonic and Form SET Rd, B		Clo
R	NS, S	1,0,1,0,0,1,0,1	Rd	b
		SET Rd↑, B		
IR	NS	0,0,1,0,0,1,0,1	Rd ≠ 0	b
IR	S	SET RRd↑, B 0,0,1,0,0,1,0,1	RRd ≠ 0	b
tiouro	NS	SET LABEL, B 0 1 1 1 1 0 1 0 1 1 0 1 1 ADDR		b .
DA	ti la se comb y	SET LABSSO, B 0 1 1 1 0 0 1 1 0 1 0 SEGMENT	0 ₁ 0 ₁ 0 ₁ 0 OFFS	b ET
	SLO	SET LABEL, B 0 1 1 1 0 0 1 0 1 1 SEGMENT OFF		b
×	NS	SET LABEL (Rx), B 0,1,1,0,0,1,0,1 ADDR		b
X	SSO	SET LABSSO (Rx), B 0 1 1 1 1 0 1 1 1 0 1 1 0 SEGMENT	Rx ≠ 0 OFFS	b ET
X	SLO	SET LABEL (Rx), B 0 1 1 1 0 0 1 0 1 1 SEGMENT OFF	Rx ≠ 0	b .

Description

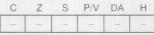
Operation
word dst <b bit>←1

The selected bit of the word destination is set to one. The remaining 15 bits are unaltered. The destination is determined by the applicable addressing mode, while the bit to be set is determined by the binary value of the b field of the instruction.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into a binary value in the b field of the instruction. The range of B is zero through 15, and b = B. Specifying a B outside of the allowable range produces an assembler error.

Flags



- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

Flags are not affected.

SET

SET bit in word (dynamic)

SET Rd, Rs

SET

Mode	Version	Mnemonic and Form SET Rd, Rs			Clocks	Operation Rd bit specified in Rs(0)	:3)> ← 1	
3	NS, S	0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 1 Rd	0,0,0,0	Rs	10	10,5,0,0,1,0,1	е ,еи	
						SET Rdf. B		
						1,0,1,0,0,1,0,0		
						1.0.1.0.0.1.0.0	8	- 8
						Description The selected bit of the waset to one. The destination is the general-purpose re	on word op egister desi	erand gnated
						by the Rd field of the inst the destination register to determined by a binary of least significant four bits	be set is lecode of the of a general	ne al-
						purpose word register de Rs field of the instruction 15 bits of the destination	. The rema	ining
						SET LABEL (EX). B. 0.1.0.1 0.1.0.1 0.1.0.1 0.1.0.1		
						8 (xR) C-2PALTSP		
						O SE IMENT		
						O 1,1,0,0,1,0,1 SEGMENT OF		
C Z	Z S P/	V DA H	Flags are	not affecte	d.15 age 21.b			
- = Una = Set	affected							

SET bit in byte (static)

SET dst, B

SETB

Mode	Version	Mnemonic and Form	Clocks	
R	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 Rbd b	0.047.0	1000
IR	NS	SET Rd↑, B 0101101011010 Rd ≠ 0 b	11	
IR	S	SET RRd↑, B 0_0_1_1_0_0_1_0_0 RRd ≠ 0 b	11	-
DA	NS	SET LABEL, B 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 b ADDRESS	13	
DA	SSO	SET LABSSO, B 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 0	14	
DA	SLO	SET LABEL, B 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0	16	
X	NS	SET LABEL (Rx), B 0 1 1 1 1 0 1 0 1 1 0 1 0 Rx ≠ 0 b ADDRESS	14	
X	SSO	SET LABSSO (Rx), B 0 1 1 1 0 0 1 0 0 Rx ≠ 0 b 0 SEGMENT OFFSET	14	
X	SLO	SET LABEL (Rx), B 0 1 1 1 0 0 1 0 0 Rx ≠ 0 b 1 SEGMENT OFFSET	17	

Description

Operation

byte dst <b bit>←1

The selected bit of the byte destination is set to one. The remaining seven bits are unaltered. The destination is determined by the applicable addressing mode, while the bit to be set to one is determined by the binary value of the b field of the instruction.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into a binary value in the b field of the instruction. The range of B is zero through 15, and b=B. Specifying a B outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

SETB

SET bit in byte (dynamic)

SETB Rbd, Rs

SETB

Mode	Version	Mnemonic and Form			Clocks	Operation Operation De (0.0)	
3	NS, S	SETB Rbd, Rs 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 Rbd	0,0,0,0	Rs	10	Rbd bit specified in Rs(0:2)><	
					0 5 68 [a,667,666 (0,011,00,011,0,0	
					0 % 688 [SET RR41, 6	2 8
						Description	
					19,0,9,0 RESS	The selected bit of the byte des set to one. The destination byte is the general-purpose register by the Rbd field of the instruction	e operand designated
					10,0,0,0	of the destination register to be determined by binary decode o significant three bits of a gener- word register designated by the	set is of the least al-purpose
					10:0:0:0 YBe	the instruction. The remaining s of the destination are unaltered	seven bits
					Ess e o	SETLABELIRA: B 0.1.d.C.O.1.0.0 00 18 ADD	
						SO . STEMPISO (TX). B. O.	
						Set DAMPH (60) THE SECONDAIN TO SECONDAIN	
Flags C	Z S P/	V DA H	Flags are i	not affecte	d. 8 898F3	H AG V9 8	, pga
-							
- = U	naffected et						



hog Ol mon see SET FLAGS

SETFLG

SET FLG LIST

lode	Version	Mnemonic and Form	Clocks	Operation FCW <c;z;s;p v="">←1</c;z;s;p>	
-	NS, S	1,0,0,0,1,1,0,1 C,Z,S,PV 0,0,0,1	b87	(see description below)	
				7809	

Description

The CPU flags, C, Z, S and P/V are set or unaltered according to the bit settings in the instruction field as described in the table below.

Instruction Bit	If = 0	If = 1	Assembler Notation
The instru	No effect	Set C flag	CY
6	No effect	Set Z flag	ZR
5	No effect	Set S flag	SGN
4	No effect	Set P/V flag	PY or OV

Assembler Notation

The assembler notation LIST refers to a list of any or all of the following reserved words, separated by commas: CY, ZR, SGN, PY, or OV. Note that PY and OV affect the same flag.

Flags

C Z S P/V DA H

* * * * - -

See description.

- = Unaffected

1 = Set

0 = Cleared

SIN

SPECIAL INPUT word to register from I/O port

SIN

SIN Rd, PORT
This is a SYSTEM instruction.

Mode Versio	SIN Rd,	PORT	Clocks	Clocks	Operation Rd <0:15>←port src<0:15>
PA NS, S	0,0,1	11110111 PORT AD	Rd 0,1 DRESS	12	G, O, F, 1, 0, 0, 0 / I 8, 2M -
					Description outlighose
					from the instruction. The original contents
					The instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST ₀ -ST ₃ .
					Assembler notation UST raters to a list of any lose in a less of any lose in a PY and OV affect the same flag.
Flags C Z S	P/V DA	н	Flags are not a	affected.	Tags C Z S PV DA H

SINB

SPECIAL INPUT byte to register from I/O port

SINB

SINB Rbd. PORT

Mode	Version	Mnemonic and Form SINB Rbd, PORT		Clocks	Operation Rbd<0:7>←port src<0:7:	landV >	
PA	NS, S	0 0 1 1 1 0 1 0 Rbd PORT ADDRESS	0,1,0,1	12	sp [0.0.0.0]		
					SIND RRdt, Rg, Rc 0,0,1,1,1,10,11, 0,0,0,10, Rc		

register designated by the Rbd field of the instruction is loaded from an input port. The port address is determined directly from the instruction. The original contents of the destination are lost.

The instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST₀-ST₃.

5

Flags

C Z S P/V DA H

Flags are not affected.

– = Unaffected

1 = Set

0 = Cleared

SIND

SPECIAL INPUT word from I/O port to memory, autodecrement

SIND

SIND dst, Rp, Rc

This is a SYSTEM instruction.

lode Version Mnemonic and F		Clocks	Operation dst<0:15>←port src<0:15>
R, PR NS 0,0,1,1,1,1,0 0,0,0,0 F	1 1 Rp 1 Rd 1	10 ₁ 0 ₁ 1 10 ₁ 0 ₁ 0	Rd<0:15>←Rd<0:15>-2 Rc<0:15>←Rc<0:15>-1
SIND RRd↑, Rp, I	1 ₁ 1 Rp 1	21	
			Description
general purpose byte destination gister designated by the Flod field of the struction is leaded from an input port to port address is determined directly an tire instruction. The original contents the designation are lost.			Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose
			register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then decremented by two. The contents of the general-purpose register designated by Rc are then decremented by one.
			This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST ₀ -ST ₃ .
			This instruction uses both indirect register memory addressing and port register port addressing modes.
			R0 can be designated as the general- purpose port source or destination register.

P/V DA Н

P/V: Set to one if the result of decrementing Rc register is zero. Reset otherwise.

= Unaffected

1 = Set

0 = Cleared

SPECIAL INPUT byte from I/O port to memory, autodecrement

SINDB

SINDB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form SINDB Rd↑, Rp, Rc	Clocks	Operation dst<0:7>←port src<0:7>
R, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 Rp 1 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc Rd 1 ₁ 0 ₁ 0 ₁ 0	21	Rd<0:15>←Rd<0:15>-1 Rc<0:15>←Rc<0:15>-1
R, PR	S	SINDB RRd1, Rp, Rc 0 0 0 1 1 1 1 1 0 1 1 0	21	SINDE REAL POLE 10.011.11.01 2.010.01 Re

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose registers designated by Rd and Rc are then decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST₀-ST₃.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose port source or destination register.

Flags

C Z S P/V DA H

P/V: Set to one if the result of decrementing Rc register is zero. Reset otherwise.

- Unaffected

1 = Set

0 = Cleared

SINDR

SPECIAL INPUT word from I/O port to memory, autodecrement and repeat

SINDR

SINDR dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form SINDR Rd↑, Rp, Rc	Clocks	Operation dst<0:15>←port src<0:15>	
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 Rp 1 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc Rd 0 ₁ 0 ₁ 0		Rd<0:15>←Rd<0:15>-2 Rc<0:15>←Rc<0:15>-1 Repeat until termination.	
IR, PR	S	SINDR RRd↑, Rp, Rc 0 0 1 1 1 0 1 1 Rp	0 ₁ 1 0 ₁ 0 11 + 10n*	SHIDS RR(1, 80 Rc 0,011,171,1011,1 0,010,010, Rc	

^{*}n is the number of iterations.

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into the memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then decremented by two. The contents of the generalpurpose register designated by Rc are then decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST₀-ST₃.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose port source or destination

Flags

S P/V DA H P/V: Set to one. 1

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

SINDRB

SINDRB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form SINDRB Rd↑, Rp, Rc			Clocks
IR, PR	NS	0,0,1,1,1,0,1,0	Rp	1,0,0,1	11 + 10n*
in, rn	NS S	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd	0,0,0,0	110 + 1011
		SINDRB RRd↑, Rp, Rc			
IR, PR	c	0,0,1,1,1,0,1,0	Rp	1,0,0,1	11 + 10n
in, rn	S	0,0,0,0 Rc	RRd	0,0,0,0	11 + 10n

Operation

dst<0:7>←port src<0:7>
Rd<0:15>←Rd<0:15>-1
Rc<0:15>←Rc<0:15>-1
Repeat until termination.

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into the memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd and Rc are then decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST₀-ST₃.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose port source or destination register.

Flags

C Z S P/V DA H

P/V: Set to one.

- = Unaffected

1 = Set

0 = Cleared

SINI

SPECIAL INPUT word from I/O port to memory, autoincrement

SINI

SINI dst, Rp, Rc

This is a SYSTEM instruction.

IR, PR	NS	SINI Rd↑, Rp, 0,0,1,1,1,1 0,0,0,0		Rp Rd	0,0,0,1	21	dst<0:15>←port src<0:15> Rd<0:15>←Rd<0:15>+2 Rc<0:15>←Rc<0:15>−1
IR, PR	S	SINI RRd1, Rp	01111	Rp	0,0,0,1	gFl 21	SINDRE RROT RP. R 0 [0,1,1,1,0,1,1, 0 [0,0,0,0] Rc
		0101010	Rc	RRd	1,0,0,0		non etil si ne
							Description
							Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction is similar in operation
							to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST ₀ -ST ₃ .
							This instruction uses both indirect register memory addressing and port register por addressing modes.
							R0 can be designated as the general- purpose port source or destination register.

Flags

C Z S P/V DA H

P/V: Set to 1 if the result of decrementing Rc register is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

SPECIAL INPUT byte from I/O port to memory, autoincrement

SINIB

SINIB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form SINIB Rd↑, Rp, Rc			Clocks	Operation dst<0:7>←port src<0:7>	
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rp Rd	0,0,0,1	21	Rd<0:15>←Rd<0:15>+1 Rc<0:15>←Rc<0:15>-1	
IR, PR	S	SINIB RRd↑, Rp, Rc 0 0 1 1 1 0 1 0 0 0 0 0 Rc	Rp RRd	0,0,0,1	21	SINIR RRdf. Ro. Ro 0.0(1,1,1,0)1,1 0.0(2,0)0 Ro	

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST₀-ST₃.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose port source or destination register.

Flags

C Z S P/V DA H

P/V: Set to 1 if the result of decrementing Rc register is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

SINIR

SPECIAL INPUT word from I/O port to memory, autoincrement and repeat

SINIR

SINIR dst, Rp, Rc

Mode	Version	Mnemonic and Form	Clocks		Clocks	Operation	
		SINIR Rd↑, Rp, Rc	Rp	0,0,0,1	69	dst<0:15>← port src<0:15> Rd<0:15>←Rd<0:15>+2	
IR, PR	NS	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd	0,0,0,0	11 + 10n*	Rc<0:15>←Rc<0:15>−1 Repeat until termination.	
		SINIR RRd↑, Rp, Rc				on grand and and	
IR, PR	S	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rp RRd	0,0,0,1	11 + 10n*	0,0,0,1,1,1,0,1,0 0,0,0,0 Rc	
		*n is the number	er of iterati	ons.			
						Description	
						Data word from the nort addresse	d by the

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by two. The contents of the generalpurpose register designated by Rc are decremented by one. This instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST₀-ST₃.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-

purpose port source or destination register. Flags C P/V: Set to 1. P/V DA - = Unaffected 1 = Set 0 = Cleared = Conditional - see description

SPECIAL INPUT byte from I/O port to memory, autoincrement and repeat

SINIRB

SINIRB dst, Rp, Rc

This is a SYSTEM instruction.

Mode IR, PR	Version	Mnemonic and Form SINIRB Rd↑, Rp, Rc 0 0 1 1 1 1 1 0 1 1 0 0 1 0 1 0 1 Rc	Rp Rd	Clocks 0 0 0 1 0 1 1 1 + 10n*	Operation dst<0:7>←port src<0:7> Rd<0:15>←Rd<0:15>+1 Rc<0:15>←Rc<0:15>-1
IR, PR	S	SINIRB RRd↑, Rp, Rc 0 0 1 1 1 0 1 0 0 0 0 0 Rc *n is the number	Rp RRd of iterati	0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Repeat until termination.
					Description
					Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one. This instruction is terminated when the result of this decrementation reaches zero.
					This instruction is interruptible. This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST ₀ -ST ₃ . This instruction uses both indirect register memory addressing and port register port addressing modes. R0 can be designated as the general-purpose port source or destination register.

C Z S P/V DA H P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

SLA

NS, S

R

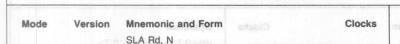
Issues bris themetonical SHIFT word arithmetic left of TU9/1 JAI0998

SLA Rd, N

1,0,0,1

13 + 3N*

SLA

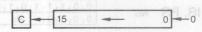


1,0,1,1,0,0,1,1

n
*N is the number of places shifted.

Rd

Operation



Description

The contents of the word destination register are shifted left. The destination is a general-purpose word register designated by the Rd field of the instruction. The number of places to be shifted is determined from the value of the field of the instruction. The magnitude of the shift must be in the range zero to 16. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 16, and n=N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	Н
*	*	a)¢	*	-	-

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Loaded from the last bit shifted out of the word register.
- Z: Set to 1 if the result is zero. Reset otherwise.
- S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
- P/V: Set to 1 if sign of register changed during shift operation. Reset otherwise.

SLAB

SHIFT byte arithmetic left

SLAB Rbd, N

SLAB

Mode	Version	Mnemonic and Form	Clocks
		SLAB Rbd, N	
D	NS. S	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 Rbd 1 ₁ 0 ₁ 0 ₁	13 + 3N*
90	N5, 5	n	13 + 314
		*N is the number of places shifted.	lacos shi

C 7 0 0

Description

Operation

The contents of the byte destination register are shifted left. The destination is a general-purpose byte register designated by the Rbd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to eight. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to eight, and n=N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

DA H C: Loaded from the last bit shifted out of the byte register.

Z: Set to 1 if result is zero. Reset otherwise.

S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.

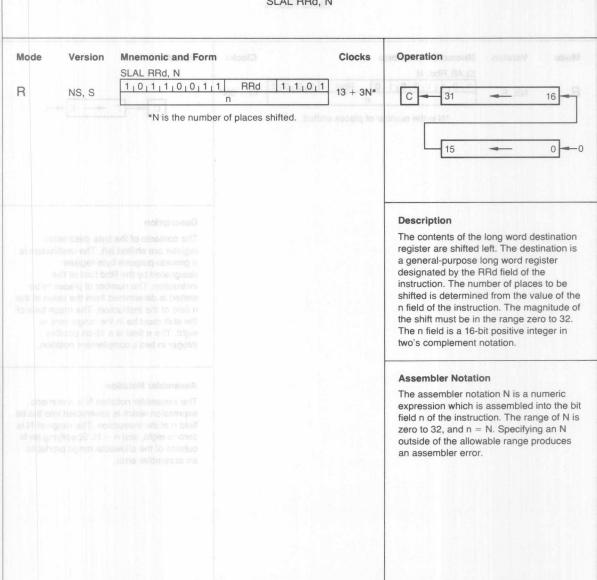
P/V: Set to 1 if sign of register changed during shift operation. Reset otherwise.

SLAL

SHIFT long word left arithmetic

SLAL RRd, N

SLAL



Flags

l	C	Z	S	P/V	DA	H
	*	*	*	*	-	-

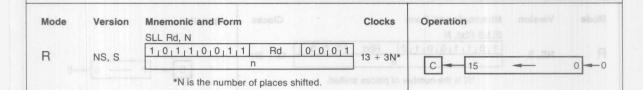
- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Loaded from the last bit shifted out of the word register.
 - Z: Set to 1 if the result is zero. Reset otherwise.
 - S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
 - P/V: Set to 1 if sign of register changed during shift operation. Reset otherwise.

SLL

SHIFT word logical left

SLL

SLL Rd. N



Description

The contents of the word destination register are shifted left. The destination is a general-purpose word register designated by the Rd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 16. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

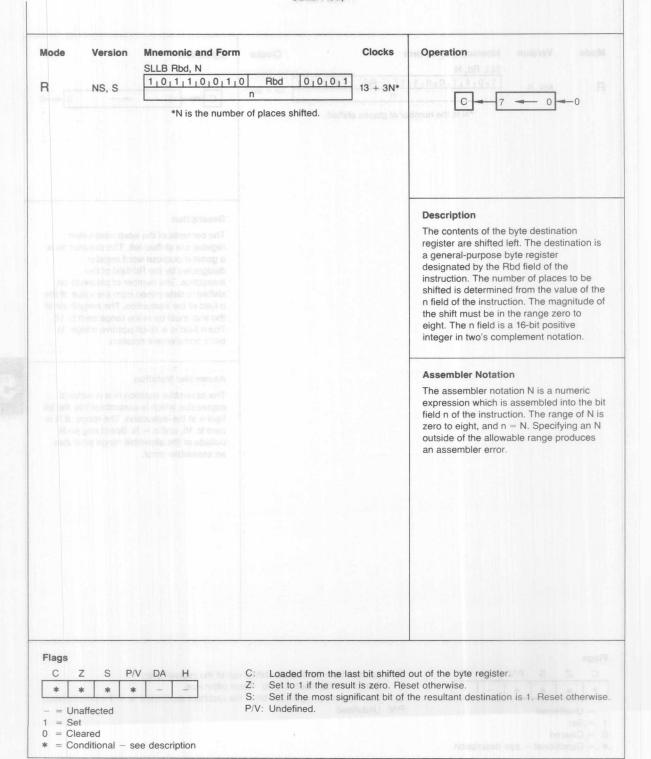
The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 16, and n=N. Specifying an N outside of the allowable range produces an assembler error.

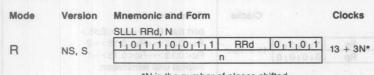
Flags



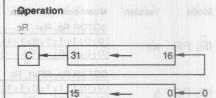
- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- H C: Loaded from the last bit shifted out of the register pair.
 - Z: Set to 1 if the result is zero. Reset otherwise.
 - S: Set if the most significant bit of the resultant destination is 1. Reset otherwise. P/V: Undefined.

5





*N is the number of places shifted.



Description

The contents of the register pair are shifted left. The register pair is designated by the RRd field of the instruction. The magnitude of the shift is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 32. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 32, and n=N. Specifying an N outside of the allowable range produces an assembler error.

Flags



- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Loaded from the last bit shifted out of the register pair.
- Z: Set to 1 if the result is zero. Reset otherwise.
- S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
- P/V: Undefined.

SOTDR

SPECIAL OUTPUT word from memory to I/O port, autodecrement and repeat

SOTDR

SOTDR Rp, src, Rc

This is a SYSTEM instruction.

lode Version	Mnemonic and Form			Clocks	Operation port dst<0:15>←src<0:15>
R, PR NS	SOTDR Rp, Rs↑, Rc 0 0 1 1 1 0 1 1 0 0 0 0 Rc	Rs Rp	1,0,1,1	11 + 10n*	Rs<0:15>←Rs<0:15>-2 Rc<0:15>←Rc<0:15>-1 Repeat until termination.
	SOTDR Rp, RRs1, Rc		.bell	of places shi	repeat until termination.
R, PR s	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rc	RRs Rp	0101010	11 + 10n*	
	*n is the numbe	r of iterati	ons.		
					Description
					A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by two. The contents of the general-purpose register
					designated by the Rc field is decremented by one. The instruction is terminated when the result of this decrementation reaches zero. This instruction is interruptible. This instruction is similar in operation
					to the corresponding standard I/O instruction. The only difference is the value on the status lines ST ₀ -ST ₃ .
					This instruction uses both indirect register memory addressing and port register por addressing modes.
					R0 can be designated as the general- purpose source or port destination register.

- - 1 - - 1

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

SOTDRB

SOTDRB Rp. src. Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form SOTDRB Rp, Rs↑, Rc			Clocks	Operat
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rp	1,0,1,1	11 + 10n*°	Rs<0: Rc<0: Repeat
IR, PR	S	SOTDRB Rp, RRs↑, Rc 0 0 1 1 1 0 1 0 0 0 0 0 0 Rc	RRs Rp	1,0,1,1	11 + 10n*	nepea 99 (and

^{*}n is the number of iterations.

Operation

port dst<0:7>←src<0:7>
Rs<0:15>←Rs<0:15>−1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

A data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by one. The contents of the general-purpose register designated by the Rc field is decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST₀-ST₃.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose source or port destination register.

Flags

C Z S P/V DA H

P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

SOTIR

SPECIAL OUTPUT word from memory to I/O port, autoincrement and repeat

SOTIR

SOTIR Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form			Clocks	Operation of the Control of the Cont	
		SOTIR Rp, Rs↑, Rc				port dst<0:15>←src<0:15>	
IR, PR	NS	0,0,1,1,1,0,1,1	Rs	0,0,1,1	11 + 10n*	Rs<0:15>←Rs<0:15>+2	
,	140	0101010 Rc	Rp	0101010	TI die	Rc<0:15>←Rc<0:15>-1	
		SOTIR Rp, RRs↑, Rc				SOTORB Ro, RRst; R	
IR, PR	S	0,0,1,1,1,0,1,1	RRs	0,0,1,1	11 + 10n*	0,1,0,1,1,1,0,0	
in, rn s	0101010 Rc	RRp	0101011	11 + 1011	5A [0,0,0,0]		

Description

A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by the Rc are decremented by one. The instruction terminates when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines ST₀-ST₃.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose source or port destination register.

Flags C Z S P/V DA H P/V: Set to 1. - - - 1 - - = Unaffected 1 = Set 0 = Cleared * = Conditional - see description

SOTIRB Rp, src, Rc

This is a SYSTEM instruction.

Rs 0,0,1,1 Rp 0,0,0,0	Clocks 11 + 10n*	Operation port dst<0:7>←src<0:7> Rs<0:15>←Rs<0:15>+1 Rc<0:15>←Rc<0:15>-1	
Rs 0 ₁ 0 ₁ 1 ₁ 1 Rp 0 ₁ 0 ₁ 0 ₁ 0	11 + 10n*		
	ecanoca	nc 0.13 ~ nc 0.13 ~ 1	
RRs 0 ₁ 0 ₁ 1 ₁ 1 RRp 0 ₁ 0 ₁ 0 ₁ 0	11 + 10n*		
-	RRs 0 0 1 1 1 1 RRp 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0		

Description

A data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by one. The contents of the general-purpose register designated by the Rc are decremented by one. The instruction terminates when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines ST₀-ST₃.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose source or port destination register.

Flags

C Z S P/V DA H

P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

SOUT

1 = Set 0 = Cleared

= Conditional - see description

SPECIAL OUTPUT word from register to I/O port

SOUT

SOUT PORT, Rs

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form SOUT PORT, Rs		Clocks	Operation port dst<0:15>←Rbs<0:15>
PA	NS, S	0,0,1,1,1,0,1,1	Rs 0,1,1,1 DDRESS	12	PR NS 0:0:0:0 Ro
		PORTA	DURESS		
					90TIRS Rp. RRSt, Rd
					PR 8 0,0,0,0 Rc
					datun erit ei n*
					Description
					The contents of the general-purpose wor
					source register designated by the Rs field of the instruction are loaded into an output
noise					port. The port address is determined
					directly from the instruction. The source contents are unaltered.
					This instruction is similar in operation to the corresponding standard I/O
					instruction. The only difference is the
					value on the status lines ST ₀₋ ST ₃ .
lags					
C Z	S P/\	V DA H	Flags are not affected.		

SOUTB

SPECIAL OUTPUT byte from register to I/O port

SOUTB

SOUTB PORT, Rbs

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form			Clocks	Operation
		SOUTB PORT, Rbs				port dst<0:7>←Rbs<0:7>
PA	NIC C	0,0,1,1,1,0,1,0	Rbs	0,1,1,1	10	Tataont Patroco
FA	NS, S	PORT ADD	RESS	MIGIGIT.	12	BR [0,0,0,0]

Description

The contents of the general-purpose byte source register designated by the Rbs field of the instruction are loaded into an output port. The port address is determined directly from the instruction. The source contents are unaltered.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST₀-ST₃.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

SOUTD

SPECIAL OUTPUT word from memory to I/O port, autodecrement

SOUTD

SOUTD Rp, src, Rc

This is a SYSTEM instruction.

SOUTD Rp, RRst, Rc O_1O_11_1 1_1O_11_1 RRs 1_1O_11_1 RRs O_1O_10_1 Rc Rp Rp O_1O_10_1 Rc Rp Rp Rp Rp Rp Rp Rp
Data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rd field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines ST ₀ -ST ₃ . This instruction uses both indirect register memory addressing and port register port
R0 can be designated as the general-

Flags

C Z S P/V DA H

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

SOUTDB

SOUTDB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form SOUTDB Rp, Rs1, Rc			Clocks	Operation port dst<0:7>←src<0:7>	
IR, PR	NS	0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rs Rp	1,0,1,1	21	Rs<0:15>←Rs<0:15>-1 Rc<0:15>←Rc<0:15>-1	
IR, PR	S	SOUTDB Rp, RRs1, Rc 0 0 1 1 1 0 1 0 0 0 0 0 Rc	RRs Rp	1,0,1,1	21	SOUTH Pp. RRST. Ro 0.0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	

Description

Data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rd field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines $\mathsf{ST}_0\text{-}\mathsf{ST}_3$.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose source or port destination register.

Flags

C Z S P/V DA H

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

SOUTI

SPECIAL OUTPUT word from memory to I/O port, autoincrement

SOUTI

SOUTI Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form SOUTI Rp, Rs↑, Rc	Rs	0,0,1,1	Clocks	Operation port dst<0:15>←src<0:15> Rs<0:15>←Rs<0:15>+2
IR, PR	NS	0 ₁ 0 ₁ 0 ₁ 0 Rc	Rp	1,0,0,0	21	Rc<0:15>←Rc<0:15>−1
IR, PR	S	SOUTI Rp, RRs1, Rc 010111110111 0101010 Rc	RRs Rp	0,0,1,1	21	SOUTDS Ro. FRat. P 0.0.11.11.0.10.0 0.0.0.0 Ro R. PR S
						Description Data word in memory, addressed by the
						contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one.
						This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST ₀ -ST ₃ .
						This instruction uses both indirect register memory addressing and port register port addressing modes.
						R0 can be designated as the general- purpose source or port destination register.

Flags

C Z S P/V DA H

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

SPECIAL OUTPUT byte from memory to I/O port, autoincrement

SOUTIB

SOUTIB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation port dst<0:7>←src<0:7>	
IR, PR	NS		0,1,1	Rs<0:15>←Rs<0:15>+1 Rc<0:15>←Rc<0:15>-1	
IR, PR	S		0111 21	redmun erti si M"	

Description

Data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST₀-ST₃.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the generalpurpose source or port destination register.

Flags

C Z S P/V DA H

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

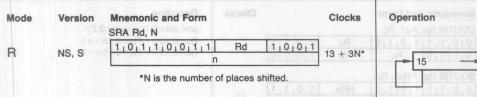
0 = Cleared

SRA

SHIFT word arithmetic right

SRA Rd. N

SRA





Description

The contents of the word destination register are shifted right. The destination is a general-purpose word register designated by the Rd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 16. The n field is a 16-bit negative integer in two's complement notation.

This operation is identical to the operation SRL apart from the treatment of the most significant bit of the word, bit 15. This bit is unaltered during the shift operation, and shifts into the adjacent bit, bit 14. Thus a signed operand has the sign preserved during the shifting operation.

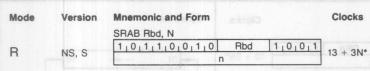
Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 16, and n=N. Specifying an N outside of the allowable range produces an assembler error.

Flags

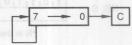
С	Z	S	P/V	DA	H
*	*	ajk	0	-	-

- = Unaffected
- 1 = Set
- 0 = Cleared
- Conditional see description
- C: Loaded from the last bit shifted out of the word register.
- Z: Set to 1 if the result is zero. Reset otherwise.
- S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
- P/V: Reset.



*N is the number of places shifted.

Operation



Description

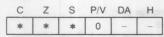
The contents of the byte destination register are shifted right. The destination is a general-purpose byte register designated by the Rbd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to eight. The n field is a 16-bit negative integer in two's complement notation.

This operation is identical to the operation SRLB apart from the treatment of the most significant bit of the byte, bit seven. This bit is unaltered during the shift operation, and shifts into the adjacent bit, bit six. Thus a signed operand has its sign preserved during the shifting operation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to eight, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags



- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Loaded from the last bit shifted out of the byte register.
- Z: Set to 1 if the result is zero. Reset otherwise.
- S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
- P/V: Reset.

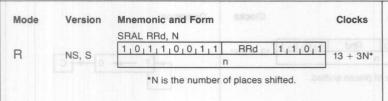
5

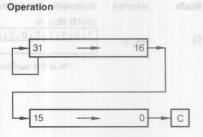
SRAL

SHIFT long word right arithmetic

SRAL RRd, N

SRAL





Description

The contents of the long word destination register are shifted right. The destination is a general-purpose long word register designated by the RRd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 32. The n field is a 16-bit negative integer in two's complement notation.

This operation is identical to the operation SRLL apart from the treatment of the most significant bit of the long word, bit 31. This bit is unaltered during the shift operation, and shifts into the adjacent bit, bit 30. Thus a signed operand has its sign preserved during the shifting operation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 32, and n=N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

* * * 0 - -

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- P/V DA H C: Loaded from the last bit shifted out of the register pair.
 - Z: Set to 1 if the result is zero. Reset otherwise.
 - S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
 - P/V: Reset.

SRL

SHIFT word logical right

SRL Rd, N

SRL

Mode	Version	Mnemonic and Form SRL Rd, N		Clocks	Operation passents and another about
R	NS, S	1,0,1,1,0,0,1,1 n	Rd 0 ₁ 0 ₁ 0	13 + 3N*	0 15 0 C

Description

The contents of the word destination register are shifted right. The destination is a general-purpose word register designated by the Rd field of the instruction. The number of places to be shifted is determined from the value of the field of the instruction. The magnitude of the shift must be in the range zero to 16. The n field is a 16-bit negative integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 16, and n=N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

P/V DA H C: Loaded from the last bit shifted out of the word register.

Z: Set to 1 if the result is zero. Reset otherwise.

S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.

P/V: Undefined.

SRLB

SHIFT byte logical right SRLB Rbd, N

SRLB

Mode	Version	Mnemonic and Form			Clocks
		SRLB Rbd, N			
D	NS. S	1,0,1,1,0,0,1,0	Rbd	0,0,0,1	13 + 3N*
H	N5, 5	n	13 + 3N*		

A PERTURA

Operation

0 - 7 - 0 C

Description

The contents of the byte destination register are shifted right. The destination is a general-purpose byte register designated by the Rbd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to eight. The n field is a 16-bit negative integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to eight, and n=N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C Z S P/V DA H

- P/V DA H C: Loaded from the last bit shifted out of the byte register.
 - Z: Set to 1 if the result is zero. Reset otherwise.
 - S: Set if the most significant bit of the resultant destination is 1. Reset otherwise. P/V: Undefined.

- = Unaffected

1 = Set

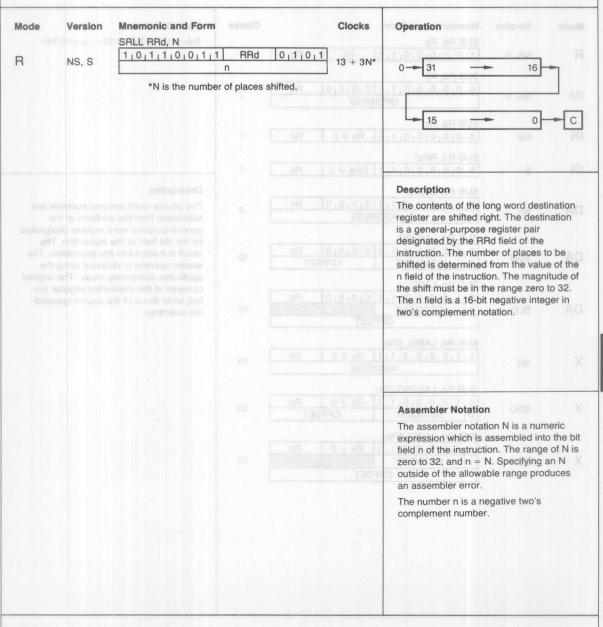
0 = Cleared

SRLL

SHIFT long word logical right (static)

SRLL

SRLL RRd. N



Flags

C	Z	S	P/V	DA	H
*	*	*	*	-	-

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Loaded from the last bit shifted out of the register pair.
- Z: Set to 1 if the result is zero. Reset otherwise.
 - S: Set if the most significant bit of the resultant destination is 1. Reset otherwise. P/V: Undefined.

SUB

SUBTRACT word from register

SUB

SUB Rd, src

Mode	Version	Mnemonic and Form	Clocks	Operation
n		SUB Rd, Rs		Rd<0:15>←Rd<0:15>-src<0:15>
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 Rs Rd	4	111010111111111111111111111111111111111
		SUB Rd, IM		
М	NS, S	0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd	2003lg to 19	idmun edi s W*
141	140, 0	OPERAND		
		SUB Rd, Rs↑		
IR	NS	$0_10_10_10_10_10_11_11$ Rs $\neq 0$ Rd	7	
		SUB Rd, RRs↑		
R	S	$0_10_10_10_10_10_11_11$ RRs $\neq 0$ Rd	7	
		SUB Rd, LABEL		Description
witeri		0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd		The source word operand contents are
DA	NS	ADDRESS	9	subtracted from the contents of the
		0110 01 1 10000		general-purpose word register designa
		0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd		by the Rd field of the instruction. The result is loaded into the destination. Tl
DA	SSO	0 SEGMENT OFFSET	10	source operand is obtained using the
		SUB Rd, LABEL		applicable addressing mode. The original
		0,1,0,0,0,0,1,1,1,0,0,0,0 Rd		contents of the destination register are
AC	SLO	1 SEGMENT	12	lost while those of the source operand are unaltered.
		OFFSET		
		SUB Rd, LABEL (Rx)		
X	NS	$0_11_10_10_10_10_11_11 \text{ Rx } \neq 0 \text{ Rd}$	10	
^	INS	ADDRESS	10	
		SUB Rd, LABSSO (Rx)		
X	SSO	$0_1 1_1 0_1 0_1 0_1 0_1 1_1 1 $ Rx $\neq 0$ Rd	10	
		0 SEGMENT OFFSET	10	
	020	1 SEGMENT	13	
		OFFSET		

Flags

C S * *

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

contents are nts of the ister designated ruction. The estination. The ed using the de. The original n register are urce operand

P/V DA H C: Reset on carry from the most significant bit of result. Set to 1 otherwise (i.e., borrow).

Z: Set to 1 if the result is zero. Reset otherwise.

S: Set to 1 if the result is negative. Reset otherwise.

P/V: Set to 1 on arithmetic overflow. Reset otherwise.

SUBTRACT byte from register

SUBB

SUBB Rbd, src

Mode	Version	Mnemonic and Form Clocks SUBB Rbd, Rbs	Operation Rbd<0:7>←Rbd<0:7> -src<0:7>
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 Rbs Rbd 4	10 10 10 10 10 11 8 8 8 8 8 8 8 8 8 8 8
		SUBB Rbd, IMb	
18.4		0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd	SUBL RRd, IME
IM	NS, S	7 OPERAND 0 7 OPERAND 0	M NS. 6 BT OPER
		SUBB Rbd, Rs↑	RE 90 at
IR	NS	$0_10_10_10_10_10_11_10$ Rs $\neq 0$ Rbd 7	SUBL RRd, Rut
		SUBB Rbd, RRs↑	10,1,0,0,1,0,0,0 EM F
IR	S	$0_10_10_10_10_10_11_10$ RRs $\neq 0$ Rbd 7	1580 580 IO IO
		SUBB Rbd, LABEL	Description
DA	NS	0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd	The source byte operand contents are
ed l		ADDRESS	subtracted from the contents of the
		SUBB Rbd, LABSSO	general-purpose byte register designated by the Rbd field of the instruction. The
DA	SSO	0 1 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1	result is loaded into the destination. The
		0 SEGMENT OFFSET	source operand is obtained using the applicable addressing mode. The original
		SUBB Rbd, LABEL	contents of the destination register are
DA	SLO	0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0	lost and those of the source operand are
		OFFSET	unaltered.
		SUBB Rbd, LABEL (Rx)	0498
X	NS	$0_1 1_1 0_1 0_1 0_1 0_1 1_1 0 Rx \neq 0 Rbd$	SUBL ARE LABEL (Fu)
^	140	ADDRESS	10,10,0,10,1,0 BN
		SUBB Rbd, LABSSO (Rx)	POGR
X	SSO	0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	SUBLIBRA, LABSSO PA
		91 THEREIN	SSO 0 SECMENT
		SUBB Rbd, LABEL (Rx)	A PARTY AND
X	SLO	0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0	Turio, ort. 0.1.01
		OFFSET	SLO 1 SEGMENT
			890

Flags

C	Z	S	P/V	DA	3 H
*	*	*	*	1	*

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description
- C: Reset on carry from the most significant bit of result. Set to 1 otherwise (i.e., borrow).
 - Z: Set to 1 if the result is zero. Reset otherwise.
- S: Set to 1 if the result is negative. Reset otherwise.
 - P/V: Set to 1 on arithmetic overflow. Reset otherwise.
 - DA: Set to 1 always.
 - H: Reset on carry from most significant bit of lower 4 bits of result. Set otherwise (i.e., borrow).

5

SUBL

SUBTRACT long word from register

SUBL

SUBL RRd, src

Mode	Version	Mnemonic and Form	Clocks	Operation
		SUBL RRd, RRs		RRd<0:31>←RRd<0:31>−src<0:31>
R	NS, S	1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 RRs RRd	8	H NS, S (1,0,0,0,0,1)
		SUBL RRd, IMR		dMI bdR SSUS
		0,0,0,1,0,0,1,0,0,0,0,0 RRd		IM NES 0,0,0,0,0,1
IM	NS, S	31 OPERAND 16	90 14	T OPERANO
		15 OPERAND 0		SUBB Rbd, Rst
		SUBL RRd, Rs↑		H NS (0,0,0,0,0,1)
IR	NS	0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 Rs ≠ 0 RRd	14	
				SUBS Rbd, RRst
IR	S	SUBL RRd, RRs↑ 0		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
in.	5	$0_10_10_11_10_10_11_10$ RRs $\neq 0$ RRd	14	Description
		SUBL RRd, LABEL		The source long word operand contents
DA	ontents of the	0,1,0,1,0,0,1,0,0,0,0,0 RRd	88390	are subtracted from the contents of the
DA	NS	ADDRESS	15	general-purpose register pair designated
		SUBL RRd, LABSSO		by the RRd field of the instruction. The result is loaded into the destination. The
DA	r novembania	0,1,0,1,0,0,1,0,0,0,0,0 RRd	0	source operand is obtained using the
DA	SSO	0 SEGMENT OFFSET	16	applicable addressing mode. The original
		CURL DRA LABEL		contents of the destination register are
		SUBL RRd, LABEL 0 1 0 1 0 0 0 0 0 0		lost while those of the source operand are unaltered.
DA	SLO	1 SEGMENT	18	unanered.
		OFFSET		SUBB REC LABEL (R
		CURL DR4 LAREL (Dr.)		
		SUBL RRd, LABEL (Rx) 0		GA SU X
X	NS	ADDRESS	16	
				SUBB Rbd, LABSSO
		SUBL RRd, LABSSO (Rx)		X 880 0 SEGMENT
X	SSO	0 1 1 0 1 1 0 1 1 0 Rx ≠ 0 RRd 0 SEGMENT OFFSET	16	TAISWS 20 TO TO
		U SEGMENT OFFSET		SUBE Rod, LABEL (P
		SUBL RRd, LABEL (Rx)		11,0,0,0,0,1,0
.,		$0_1 1_1 0_1 1_1 0_1 0_1 1_1 0 Rx \neq 0 RRd$		X SLO I SEGMENT
X	SLO	1 SEGMENT	19	
		OFFSET		

Flags

C	Z	S	P/V	DA	BH
*	*	*	*	-	-

- C: Reset on carry from the most significant bit of result. Set to 1 otherwise (i.e., borrow).
 - Z: Set to 1 if the result is zero. Reset otherwise.
- S: Set to 1 if the result is negative. Reset otherwise.
- P/V: Set to 1 on arithmetic overflow. Reset otherwise.

- = Unaffected

- 1 = Set
- 0 = Cleared 1988 illustrate and 4 toworks and magnificant
- * = Conditional see description

TCC

TEST condition codes and set a bit in word

TCC

TCC CC, Rd

Mode	Version	Mnemonic and Form TCC CC, Rd		Clocks	Operation Rd <bit 0="">←1 if cond</bit>	lition is met.	
R	NS, S	1 ₁ 0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 Rd	CC	5	11:1:1:0:1:0:1		

Description

The contents of the flags are compared with those specified by the CC field of the instruction. If the comparison is successful, the least significant bit of the destination word register Rd is set to one. Otherwise this bit is unaffected. Remaining bits of the destination are not altered.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

TCCB

TEST condition codes and set a bit in byte

TCCB

TCCB CC, Rbd

Mode	Version	Mnemonic and Form	Clocks	177.39		
		TCCB CC, Rbd				1
R	NS, S	1,0,1,0,1,1,1,0	Rbd	CC	5	

Operation

Rbd
bit 0>←1 if condition is met.

Description

The contents of the flags are compared with those specified by the CC field of the instruction. If the comparison is successful, the least significant bit of the destination byte register Rbd is set to one. Otherwise this bit is unaffected. Remaining bits of the destination are not altered.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected

1 = Set

0 = Cleared

TEST dst

Mode R	Version NS, S	Mnemonic and Form TEST Rd 1 0 0 0 1 1 0 1 Rd 0 1 0 0	Clocks	Operation dst<0:15>←dst<0:15> ∨ 0
11	143, 3	[1]0 0 0 1 1 0 1 Nu 0 1 0 0		101(1114(4)01)
		TEST Rd↑		ton etest
IR	NS	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 1 Rd 0 ₁ 1 ₁ 0 ₁ 0	8	0,0,1,1,0,0,0,0,0] an Al
IR	S	TEST RRd↑ 0,0,0,0,1,1,0,1 RRd 0,1,0,0	8	16878 RRdt
		TEST LABEL		Description TRAF
	NS	0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 ADDRESS	11	The contents of the destination word operand are tested to set the appropriate flags. Testing is done by performing a
DA	SSO	TEST LABSSO 0 1 1 0 1 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0	12	logical OR operation between destination word and zero. The destination is determined by the applicable addressing
DA	SLO	TEST LABEL 0 1 1 0 1 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0	14	mode and the contents of the destination are not altered. In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.
X	NS	TEST LABEL (Rx) 0 1 0 0 1 1 0 1 Rx \neq 0 0 1 0 0 ADDRESS	12	AAN JABAN BIRAT (10,11,10,0,11,0)
X	SSO	TEST LABSSO (Rx) 0 1 1 0 1 1 1 1 0 1 Rx ≠ 0 0 1 1 0 1 0 0 SEGMENT OFFSET	12	1687B LABSEO (Rx) X 680 0 11,0 (1,1 1,0 1) SEGMENT
X	SLO	TEST LABEL (Rx) 0 1 1 0 1 0 1 1 1 1 0 1	15	TEST8 LABEL (RX) 1 0,1 1,0,1 1,0,1 X SLO 1 SEGMENT

Flags

С	Z	S	P/V	DA	H
-	*	*	-	eż <u>iv</u> mi	nil <u>o</u> ti

- Set to 1 if the result is zero. Reset otherwise.

 Set to 1 if the result is negative. Reset otherwise.
- = Unaffected1 = Set
- 0 = Cleared
- * = Conditional see description

TESTB

TEST byte

TESTB dst

TESTB

Mode R	Version NS, S	Mnemonic and Form TESTB Rbd 1,0,0,0,1,1,1,0,0 Rbd 0,1,0,0	Clocks 7	Operation dst<0:7>←dst<0:7> ∨ 0	
	,				
		TESTB Rd↑		TEST Ret	
IR	NS	0_0_0_0_1_1_1_0_0 Rd 0_1_1_0_0	8	1,0,1,1,0,0,0,0) SM:	
IR	S	TESTB RRd↑ 0_0_0_0_0_1_1_1_0_0 RRd 0_1_1_0_0	8	0,1,1,0,0,0,0	я
		TESTB LABEL		Description	
DA	NS	0,1,0,0,1,1,0,0,0,0,0,0,0,1,0,0 ADDRESS	112880	The contents of the destination byte operand are tested to set the approflags. Testing is done by performing	priate
DA	SSO	TESTB LABSSO 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 1 0	12	logical OR operation between destiin byte and zero. The destination is determined by the applicable address mode and the contents of the destin	nation
DA	SLO	TESTB LABEL 0 1 0 0 1 1 1 0 0 0	14	are not altered. In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.	
X	NS	TESTB LABEL (Rx) 0 1 1 0 1 0 1 1 1 0 0 Rx ≠ 0 0 1 1 0 1 0 ADDRESS	12	TEST LABEL (Rx) 0.11,0,0.1,1,0.1 AC	
×	SSO	TESTB LABSSO (Rx) 0	12	TEST LABSSO (Rx) 0.110.0, 1,110.0 0.000000000000000000000000000000	
		TESTB LABEL (Rx) 0 1 1 0 1 0 1 1 1 1 0 0 Rx ≠ 0 0 1 1 0 1 0		TESTLABEE (Rx)	
X	SLO	1 SEGMENT OFFSET	15	TUSMO38 11 OJ8	

Flags

С	Z	S	P/V	DA	H
_	*	*	*	-020	media

- Z: Set to 1 if the operand is zero. Reset otherwise.
- S: Set to 1 if the operand is negative. Reset otherwise.
 - P/V: Set to 1 if parity of operand is even. Reset otherwise.

- = Unaffected

- 1 = Set 0 = Cleared
- * = Conditional see description

TEST long word

TESTL

TESTL dst

Mode	Version	Mnemonic and Form TESTL RRd			Clocks	Operation dst<0:31>←dst<0:31> ∨ 0
R	NS, S	1,0,0,1,1,1,0,0	RRd	1,0,0,0	13	0.0.0.1.1.1.0.1
						0,0,0,0 Rc
						TRDS BRdt BRet, Bo
						0,0,0,1,1,1,0,1
		TESTL Rd↑		fold fold	PRS	pR 0,0,0,0
IR	NS	0,0,0,1,1,1,0,0	Rd	1,0,0,0	13	
		TESTL RRd↑				
IR	S	0,0,0,1,1,1,0,0	RRd	1,0,0,0	13	
		TESTL LABEL				Description
		0,1,0,1,1,1,0,0	0.0.0.0	0 1 1 0 1 0 1 0		The contents of the long work
	NS	ADDF	RESS	01.101010	16	are tested to set the appropri
		TESTI LABORE				Testing is done by performing
		TESTL LABSSO 0,1,0,1,1,1,0,0	0 0 0 1	0 1 1 0 0 0		OR operation between destin
DA	SSO	0 SEGMENT		FSET	17	zero. The destination is deter
		the address of a byts.	-			the applicable addressing mo
		TESTL LABEL				altered.
DA	SLO	0 1 0 1 1 1 0 0 1 SEGMENT	0,0,0,	0 1 0 0 0	19	In the IR mode, R0 (or RR0) of
DA		OFF		***************************************	19	designated as the general-pu
		O CONTROLLED OF THE PROPERTY O	-			destination register.
		TESTL LABEL (Rx)		1		
X	NS	0 ₁ 1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₀ 0 ADDF	Rx ≠ 0	1,0,0,0	17	
		, , , , ,	1E33			
		TESTL LABSSO (Rx)				
X	SSO	0,1,0,1,1,1,0,0	Rx ≠ 0		17	
- 0	H BITTY DESI	0 SEGMENT	OF	FSET		
		TESTL LABEL (Rx)				
		10 1 0 1 1 1 0 0	Rx ≠ 0	1,0,0,0		
	SLO	1 SEGMENT			20	
X		d bedenoises where OFF				

s of the long word destination set the appropriate flags. one by performing a logical n between destination and estination is determined by le addressing mode and the the destination are not

de, R0 (or RR0) can be as the general-purpose egister.

Flags

C Z S P/V DA * *

- Z: Set to 1 if the result is zero. Reset otherwise.
- S: Set to 1 if the result is negative. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

TRDB

TRANSLATE byte, autodecrement

TRDB

TRDB dst, src, Rc

Mode	Version	Mnemonic and Form TRDB Rd↑, Rs↑, Rc			Clocks	Operation (see description below)
R	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd Rs	1,0,0,0	25	(see description below)
		TRDB RRd↑ RRs↑, Rc				
R	S	1 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	RRd RRs	1,0,0,0	25	ton Jirean
						1,0,1,1,1,0,0,0) - SM
						Description 0
						The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.
		OR operation between zero. The destination the applicable address contents of the dealing altered				The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated.
						The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated.
						A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte
						is read from this address, and is loaded into the address specified by the Rd register.
						The address specified by the Rd register is decremented by one to point to the nex byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated. This completes one iteration.
						This instruction terminates after one iteration. It is a special case of the instruction TRDRB. The contents of register RH1 are undefined following TRDB.
						R0 can be designated as the general- purpose source designation register.

Flags

С	Z	S	P/V	DA	Н
-	*	-	*	-	-

Z: Undefined.

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set 0 = Cleared
- * = Conditional see description

TRDRB

TRANSLATE byte, autodecrement and repeat

TRDRB

TRDRB dst, src, Rc

Mode	Version	Mnemonic and Form TRDRB Rd↑, Rs↑, Rc		Clocks	Operation (see description below)	
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 Rd 0 ₁ 0 ₁ 0 ₁ 0 Rc Rs	0,0,0,0	11 + 14n*	10,0,0,1,1,1,0,1,1,0,10,10,10,10,10,10,1	
IR	S	TRDRB RRd1, RRs1, Rc 1,0,1,1,1,1,0,0,0 RRc 0,0,0,0,0 Rc Rs	1 ₁ 1 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0	11 + 14n*	TRIS RRdt, RRst, Rt (1 0,1,1,1,0,0,0) (0 0,0,0)	

^{*}n is the number of iterations.

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the address specified by the Rd register.

The address specified by the Rd register is decremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated. This completes one iteration.

This instruction repeats until the contents of the Rc register reach zero, indicating that the string has been exhausted. This instruction is interruptible at the end of each iteration. The contents of register RH1 are undefined following TRDRB.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H

Z: Undefined. P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

TRIB

TRANSLATE byte, autoincrement

TRIB, dst, src, Rc

TRIB

Mode	Version	Mnemonic and Form TRIB Rd↑, Rs↑, Rc			Clocks	Operation (see description below)	6005
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd Rs	0,0,0,0	25	NS (0,0,0,1,1,0,0)	
IR	s	TRIB RRd↑, RRs↑, Rc 1,0,1,1,1,1,0,0,0 0,0,0,0 Rc	RRd RRs	0101010	25	TRDRS RRdf, RRsf; 1,0,1,1,1,0,0, 0,0,0,0 Ro	

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the address specified by the Rd register.

The address specified by the Rd register is incremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated. This completes one iteration.

This instruction terminates after one iteration. It is a special case of the instruction TRIRB. The contents of register RH1 are undefined following TRIB.

R0 can be designated as the generalpurpose source or destination register.

Flags

С	Z	S	P/V	DA	Н
-	*	-	*	-	-

- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

Z: Undefined.

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

TRIRB

TRANSLATE byte string, autoincrement and repeat

TRIRB

TRIRB dst, src, Rc

Mode	Version	Mnemonic and Form TRIRB Rd↑, Rs↑, Rc			Clocks	Operation (see description below)		
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd Rs	0111010	11 + 14n*	0,0,0,1,1,1,0,0 0,0,0,0 0,0,0,0		
IR	S	TRIRB RRd↑, RRs↑, Rc 1,0,1,1,1,1,0,0,0 0,0,0,0 Rc	RRd RRs	0111010	11 + 14n*	8 1884 (588 SCIRT 9 (5) (7) (7) (8) (5) (5)	-8	aı

^{*}n is the number of iterations.

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the address specified by the Rd register.

The address specified by the Rd register is incremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated. This completes one iteration.

This instruction repeats until the contents of the Rc register reach zero, indicating that the string has been exhausted. This instruction is interruptible at the end of each iteration. The contents of register RH1 are undefined following TRIRB.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V DA H

H Z: Undefined.
P/V: Set to 1.

- = Unaffected

1 = Set

0 = Cleared

TRTDB

TRANSLATE AND TEST byte, autodecrement

TRTDB

TRTDB dst, src, Rc

Mode	Version	Mnemonic and Form TRTDB Rd↑, Rs↑, Rc			Clocks	Operation (see description below)
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd Rs	1 ₁ 0 ₁ 1 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0	25	0.0.8.7.7.8.0.0
IR	S	TRTDB RRd↑, RRs↑, Rc 1,0,1,1,1,1,0,0,0 0,0,0,0 Rc	RRd RRs	1,0,1,0	25	7,1689,1631,1831,19 2,0,1,1,1,0,0,0 0,0,0,0

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated and tested.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated and tested.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the general-purpose byte register RH1 for testing.

The address specified by the Rd register is decremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated and tested. This completes one iteration.

This instruction terminates after one iteration. It is a special case of the instruction TRTDRB.

R0 can be designated as the generalpurpose source or destination register.

Flags

C Z S P/V D A H

Z: Set to 1 if the translated byte is zero. Reset otherwise.

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

TRTDRB

TRANSLATE AND TEST byte, autodecrement and repeat

TRTDRB

TRTDRB dst, src, Rc

Mode	Version	Mnemonic and Form TRTDRB Rd↑, Rs↑, Rc			Clocks	Operation (see description below)
IR	NS	1 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rc	Rd Rs	1,1,1,0	11 + 14n*	
IR	S	TRTDRB RRd↑, RRs↑, F	RRd RRs	1,1,1,0	11 + 14n*	6 1610 1111 PART PART PART PART PART PART PART PART

^{*}n is the number of iterations.

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated and tested.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated and tested.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the general-purpose byte register RH1 for testing.

The address specified by the Rd register is decremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated and tested. This completes one iteration.

The instruction repeats until the value loaded into the RH1 register is non-zero or until the contents of the Rc register reach zero, indicating that the string has been exhausted. This instruction is interruptible at the end of each iteration.

R0 can be designated as the generalpurpose source or destination register.

Flags

С	Z	S	P/V	DA	H
-	*	of when	*	SEL_O	US_B

Z: Set to 1 if the translated byte is zero. Reset otherwise.

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

- = Unaffected

1 = Set

0 = Cleared

TRTIB

TRANSLATE AND TEST byte, autoincrement

TRTIB

TRTIB dst, src, Rc

Mode	Version	Mnemonic and Form TRTIB Rd↑, Rs↑, Rc			Clocks	Operation (see description below)
IR	NS	1,0,1,1,1,0,0,0 0,0,0,0 Rc	Rd Rs	0,0,1,0	25	0,0,0,1,0,1,0,0,0 0,0,0,1,0,1,0,0
IR	s	TRTIB RRd↑, RRs↑, Rc 1,0,1,1,1,1,0,0,0 0,0,0,0 Rc	RRd RRs	0,0,1,0	25	1,0,0,0,1,1,0,0,0 1,0,0,0,1,1,0,0,0 1,0,0,0,0

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated and tested.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated and tested.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the general-purpose byte register RH1 for testing.

The address specified by the Rd register is incremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated and tested. This completes one iteration.

This instruction terminates after one iteration. It is a special case of the instruction TRTIRB.

R0 can be designated as the generalpurpose source or destination register.

Flags

С	Z	S	P/V	DA	H
_	*	1911/2010	*	SA_DO	94-8

Z: Set to 1 if the translated byte is zero. Reset otherwise.

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

- Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

TRTIRB

TRANSLATE AND TEST byte string, autoincrement and repeat

TRTIRB

TRTIRB dst, src, Rc

Mode	Version NS	Mnemonic and Form Clocks TRTIRB Rd↑, Rs↑, Rc 1 0 1 1 1 1 0 0 0 Rd 0 1 1 1 0 0 0 Rd 0 0 0 0 0 0 0 0 Rc 1 1 1 1 1 1 0 0 0 Rc 1 1 1 1 1 1 0 0 0 0 Rc 1 1 1 1 1 1 1 0 0 0 0 Rc 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	Operation (see description below)
IR	s	TRTIRB RRd1, RRs1, Rc 1 0 1 1 1 1 0 0 0 RRd 0 1 1 1 0 0 1 0 1 0 1 0 RC RRS 1 1 1 1 1 0	

^{*}n is the number of iterations.

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated and tested.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated and tested.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address and is loaded into the general-purpose byte register RH1 for testing.

The address specified by the Rd field is incremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated and tested. This completes one iteration.

This instruction repeats until the value loaded into the RH1 register is non-zero, or until the contents of the Rc register reach zero, indicating that the string has been exhausted. This instruction is interruptible at the end of each iteration.

R0 can be designated as the generalpurpose source or destination register.

Flags

С	Z	S	P/V	DA	H
-	*	-	aje	-	-

Z: Set to 1 if the table entry is zero. Reset otherwise.

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

TSET

TEST word and set

TSET dst

TSET

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	TSET Rd [1,0,0,0,1,1,1,0,1] Rd [0,1,1,0]	7	If dst<0:15>←is negative, then S flag←1; otherwise S flag←0.
				dst<0:15>←FFFF
IR	NS	TSET Rd↑ 0,0,0,0,1,1,1,0,1 Rd 0,1,1,0	11	*n is the numb
	siye salelpa	TSET RRd↑ 010101011111011 RRd 011110	11	
antra	Iras notous	TOET LABEL		Description
DA	NS	0,1,0,0,1,1,0,1,0,0,0,0,0,1,1,1,0 ADDRESS	14	The most significant (sign) bit of the destination word is loaded into the S flag.
DA	SSO	TSET LABSSO 0 1 1 0 0 1 1 1 0 1 0 0 0 0 1 1 1 0 0 SEGMENT OFFSET	15	The contents of the destination are then set to all ones. The destination is determined by the applicable addressing mode.
DA	SLO	TSET LABEL 0 1 1 0 1 0 1 1 1 0 1 0 0 0 0 0 1 1 1 0 1 SEGMENT OFFSET	17	In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.
geni	NS	TSET LABEL (Rx) 0 1 0 0 1 1 0 1 Rx ≠ 0 0 1 1 0 ADDRESS	15	
X	SSO	TSET LABSSO (Rx) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	15	
		TSET LABEL (Rx) 0 1 1 0 1 0 1 1 1 1 0 1	18	

Flags

C Z S P/V DA H

P/V DA H S: Set to 1 if the most significant bit of the destination is one. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

Mode	Version	Mnemonic and Form TSETB Rbd	Clocks	Operation If dst<0:7> is negative,	
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 Rbd 0 ₁ 1 ₁ 1 ₁ 0	7	then S flag←1; otherwise S flag←0.	
				dst<0:7>←FF	
		TSETB Rd↑		teR bB RoX	
IR	NS	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 0 ₁ 0 Rd 0 ₁ 1 ₁ 1 ₁ 0	11	(8,0,1,0,0,0,0)	
		TSETB RRd↑		SERRING ROLL	
IR	S	0,0,0,0,1,1,0,0 RRd 0,1,1,0	11	10. B. 4. b. 2. 0. 41	2
		TSETB LABEL		Description	
DA	NS	0,1,0,0,1,1,0,0,0,0,0,0,0,1,1,0 ADDRESS	14	The most significant (sign) bit destination byte is loaded into	the S flag.
		TSETB LABSSO		The contents of the destination set to all ones. The destination	
DA	SSO	0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 SEGMENT OFFSET	15	determined by the applicable mode.	addressing
		TSETB LABEL		In the IR mode, R0 (and RR0)	
DA	SLO	0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 1 SEGMENT OFFSET	17	designated as the general-pur destination register.	
		The College of the Co			
X	NS	TSETB LABEL (Rx) 0	15	(A) LESAJ DE SOX	
				KOR Rut LABSSO (R	
X	SSO	TSETB LABSSO (Rx) $0_11_10_10_11_10_10_1 \text{ Rx} \neq 0_10_11_10_1$	15	10-0-1-0-0-1-0	
^	330	0 SEGMENT OFFSET	0 13	THE O SECRET	
		TSETB LABEL (Rx)		XON PULLABEL SEX	
X	SLO	0,1,0,0,1,1,0,0 Rx ≠ 0 0,1,1,1,0 1 SEGMENT	18	1010:1:0:0:1:0 10:0:0:1:0:0:0:0:0:0:0:0:	
	OLO .	OFFSET			





P/V DA H Some S: Set to 1 if the most significant bit of the destination is 1. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

XOR

EXCLUSIVE OR word with register

XOR

XOR Rd, src

Mode	Version	Mnemonic and Form	Clocks	Operation and a second state of the second s
R	NS, S	XOR Rd, Rs 1,0,0,1,1 Rs Rd	4	Rd<0:15>←src<0:15> ⊕ Rd<0:15>
1	143, 3	otherwise Sitaq≠−0.	0017 13	10101110101011
		XOR Rd, IM		
M	NS, S	0,0,0,0,1,0,0,1,0,0,0,0 Rd OPERAND	7	
		XOR Rd, Rs↑		
R	NS	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 Rs ≠ 0 Rd	597	10,11,19,0,0,0 SM R
IR	S	XOR Rd, RRs↑ 0_0_0_0_0_1_1_0_0_1 RRs ≠ 0 Rd	7	THE PROPERTY OF THE PROPERTY O
				Description
		XOR Rd, LABEL 0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0	100000	A logical EXCLUSIVE OR operation is
	NS	ADDRESS	9	performed between corresponding bits of
		XOR Rd, LABSSO		the source and destination words. The source operand is obtained by the
ο Δ	SSO	0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd	10	appropriate addressing mode, and the
DA	550	0 SEGMENT OFFSET	10	destination operand is always a
		XOR Rd, LABEL		general-purpose word register designated
				by the Bd field of the instruction. The
D.4	astodnind-rateu	0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 0 ₁ 0 ₁ 0 ₁ 0 Rd	1,0,0,010	by the Rd field of the instruction. The 16-bit result is loaded into the
DA	SLO	0 1 1 0 1 0 1 0 1 0 0 0 0 0 Rd 1 SEGMENT	12	16-bit result is loaded into the destination, whose original contents are
DA	SLO	0 1 1 0 1 0 1 0 1 0 1 0 0 0 0 Rd 1 SEGMENT OFFSET	12	16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.
		0 1 1 0 1 0 1 0 1 0 1 0 0 0 Rd 1 SEGMENT OFFSET XOR Rd, LABEL (Rx)	1987	16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.
	SLO	0 1 1 0 1 0 1 0 1 0 1 0 0 0 0 Rd 1 SEGMENT OFFSET	12	16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.
		0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	1987	16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.
×	NS	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	10	16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.
×		0 1 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 Rd 1 SEGMENT OFFSET XOR Rd, LABEL (Rx) 0 1 1 0 1 0 1 1 0 1 0 1 Rx ≠ 0 Rd ADDRESS XOR Rd, LABSSO (Rx)	1987	16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.
X	NS	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	10	16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.
×	NS SSO	0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	10	16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.
X	NS	0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	10	16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.
DA X X	NS	0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	10	16-bit result is loaded into the destination, whose original contents lost. The contents of the source are altered.

Flags

C	Z	S	P/V	DA	H
-	*	*	-	-	-

- Z: Set to 1 if the result is zero. Reset otherwise.
 - S: Set to 1 if the result is negative. Reset otherwise.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

XORB

EXCLUSIVE OR byte with register

XORB

XORB Rbd, src

Mode	Version	Mnemonic and Form	Clocks
do mina		XORB Rbd, Rbs	
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 Rbs Rbd	4
		XORB Rbd, IMb	
IM	NS, S	0 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd	7
	, 0	7 OPERAND 0 7 OPERAND 0	
		XORB Rbd, Rs↑	
IR	NS	$0_10_10_10_11_10_10_10$ Rs $\neq 0$ Rbd	7
		XORB Rbd, RRs↑	
IR	S	$0_10_10_10_11_10_10_10$ RRs $\neq 0$ Rbd	7
		XORB Rbd, LABEL	
DA	NS	0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd	9
DA	INS	ADDRESS	9
		XORB Rbd, LABSSO	
DA	SSO	0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 Rbd	10
	330	0 SEGMENT OFFSET	
		XORB Rbd, LABEL	
DA		0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 0 ₁ 0 ₁ 0 ₁ 0 Rbd	
DA	SLO	1 SEGMENT OFFSET	12
		XORB Rbd, LABEL (Rx)	
X	NS	0 ₁ 1 ₁ 0 ₁ 0 ₁ 1 ₁ 0 ₁ 0 ₁ 0 Rx ≠ 0 Rbd ADDRESS	10
		XORB Rbd, LABSSO (Rx) 0	
X	SSO	0 SEGMENT OFFSET	10
		VODE Phd I AREL (Pv)	
		XORB Rbd, LABEL (Rx) 0	
X	SLO	1 SEGMENT	13
		OFFSET	

Description

Operation

Rbd<0:7>←src<0:7> ⊕ Rbd<0:7>

A logical EXCLUSIVE OR operation is performed between corresponding bits of the source and destination bytes. The souce operand is obtained by the appropriate addressing mode, and the destination operand is always a general-purpose byte register designated by the Rbd field of the instruction. The 8-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.

Flags

С	Z	S	P/V	DA	Н
-	*	ak:	*	-	711

- = Unaffected

1 = Set

0 = Cleared

* = Conditional - see description

Z: Set to 1 if the result is zero. Reset otherwise.

P/V: Set to 1 if parity of result is even. Reset otherwise.

5.8 EXTENDED PROCESSING INSTRUCTIONS

The following pages include "templates" for the AmZ8001 and AmZ8002 extended processing instructions. These templates correspond to Extended Processing Architecture (EPA) instructions, which combine Extended Processing Unit (EPU) operations with possible transfers between memory and an EPU, between CPU registers and EPU registers, and between the flag byte of the CPU's FCW and the EPU.

Each of these templates is described on the following pages. The description assumes that the EPE control bit in the CPU's FCW has been set to 1. In addition, the description is from the point of

view of the CPU - that is, only CPU activities are described; the operation of the EPU is implied, but the full specification of the instruction depends upon the implementation of the EPU and is beyond the scope of this manual.

Fields ignored by the CPU are shaded in the diagrams of the templates. The 2-bit field in bit positions 0 and 1 of the first word of each template would normally be used as an identification field for selecting one of up to four EPUs in a multiple EPU system configuration. Other shaded fields would typically contain opcodes for instructing an EPU as to the operation it is to perform in addition to the data transfer specified by the template.

	600,0000000000000000000000000000000000	

LOAD Memory from EPU This is an EXTENDED instruction. Operation Version Mnemonic and Form Clocks Mode $memory \leftarrow EPU$ $0_10_10_10_11_11_11$ Rd $\neq 0$ 1_11 IR NS 11 + 3n $0_10_10_10_11_11_11$ RRd $\neq 0$ 1_11 IR S 11 + 3n0,1,0,0,1,1,1,1,0,0,0,0,1,1 DA NS 15 + 3nn - 1 ADDRESS Description The CPU performs the indicated address 0,1,0,0,1,1,1,1,0,0,0,0,1,1 calculation and generates n EPU memory DA 15 + 3nn - 1 write transactions. The n words are SEGMENT **OFFSET** supplied by an EPU and are stored in n consecutive memory locations starting with the effective address. 0,1,0,0,1,1,1,1,0,0,0,0,1,1 DA SLO 18 + 3nSEGMENT **OFFSET** $0_11_10_10_11_11_11_1$ Rx $\neq 0$ 1,1 X NS 14 + 3n**ADDRESS** $0_11_10_10_11_11_11_1$ Rx $\neq 0$ 1 1 1 X SSO 15 + 3nn-1SEGMENT **OFFSET** 1 1 $0_11_10_10_11_11_11$ Rx $\neq 0$ n – 1 X SLO 17 + 3nSEGMENT **OFFSET** Flags P/V DA Flags are not affected - = Unaffected 1 = Set

0 = Cleared

LOAD EPU from memory This is an EXTENDED instruction. **Mnemonic and Form** Clocks Operation Mode Version EPU ← memory $0_10_10_10_11_11_11_1$ Rs $\neq 0$ 0_11 IR NS 11 + 3n $0_10_10_10_11_11_11$ RRs $\neq 0$ 0_11 IR S 11 + 3n0,1,0,0,1,1,1,1,0,0,0,0,0,1 DA 15 + 3nNS n - 1 **ADDRESS** Description The CPU performs the indicated address 0,1,0,0,1,1,1,1,0,0,0,0,0 calculation and generates n EPU memory DA SSO 15 + 3nread transactions. The n consecutive SEGMENT OFFSET words are fetched from the memory locations starting with the effective address. The data is read by an EPU and 0,1,0,0,1,1,1,1,0,0,0,0,0,1 operated upon according to the extended DA 18 + 3n SLO processing instruction encoded into the SEGMENT shaded fields. **OFFSET** $0_11_10_10_11_11_11_1$ Rx $\neq 0$ 0,1 X NS 14 + 3nn-1ADDRESS 0 1 $0_11_10_10_11_11_11_1$ Rx $\neq 0$ X SSO 15 + 3nn -SEGMENT **OFFSET** $0_{1}1_{1}0_{1}0_{1}1_{1}1_{1}1_{1}1$ Rx $\neq 0$ $0_{1}1$ X SLO 17 + 3nSEGMENT **OFFSET** Flags C Z S P/V DA Flags are not affected - = Unaffected 1 = Set 0 = Cleared * = Conditional - see description

LOAD EPU from CPU

This is an EXTENDED instruction.

Mode	Version	Mnemonic and Form	Clocks	EPU← CPU registers	
R	NS, S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 1 0 1 ₁ 0 src n-1	11 + 3n	10,0,0,0,0,0,0	

Description

The contents of n words are transferred to an EPU from consecutive CPU registers starting with register src. CPU registers are transferred consecutively, with register zero following register 15.

Flags

C Z S P/V DA H

Flags are not affected.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

LOAD EPU from FCW This is an EXTENDED instruction. Clocks Operation Version **Mnemonic and Form** Mode EPU ← flags 1,0,0,0,1,1,1,1,0 R NS, S 14 0,0,0,0 Description The flags in the CPU's FCW are transferred to an EPU on address data lines AD₀-AD₇. Flags Flags are not affected. - = Unaffected 1 = Set 0 = Cleared * = Conditional - see description

Internal EPU operation This is an EXTENDED instruction. Operation Apple 10 Clocks Mode Version **Mnemonic and Form** internal EPU operation 1,0,0,0,1,1,1,0 0,1 R NS. S 14 Description The CPU treats this as a No Op. It is typically used to initiate an internal EPU operation. Flags C DA Flags are not affected. - = Unaffected 1 = Set 0 = Cleared * = Conditional - see description

LOAD CPU from EPU This is an EXTENDED instruction. Mnemonic and Form Operation Tolera Mode Version Clocks CPU ← EPU registers 1,0,0,0,1,1,1,1 R NS. S 11 + 3n dst Description The contents of n words are transferred from an EPU to consecutive CPU registers starting with register dst. CPU registers are transferred consecutively, with register zero following register 15. Flags P/V DA Flags are not affected. - = Unaffected 1 = Set 0 = Cleared * = Conditional - see description

LOAD FCW from EPU

This is an EXTENDED instruction.

Mode	Version	Mnemonic and Form	Clocks
R	NS. S	1 ₁ 0 ₁ 0 ₁ 0 ₁ 1 ₁ 1 ₁ 1 ₁ 0 0 ₁ 0	14
11	143, 3	0,0,0,0	

Operation

 $\mathsf{flags} \leftarrow \mathsf{EPU}$

Description

The flags in the CPU's FCW are loaded with information from an EPU on address lines AD₀-AD₇.

The contents of CPU register zero are undefined after the execution of this instruction.

Flags

C Z S P/V DA H

See description.

- = Unaffected
- 1 = Set
- 0 = Cleared
- * = Conditional see description

LOAD FOW from EPU This is an EXTENDED natraction. This is an EXTENDED natraction. Chocks Operation Machine Version Machine and Form Chocks Operation (lags = EPU (lags = E

nottonoes0

The liegs in the CPU's FCW are loaded with incomplice from an EPU on address lines ADo-ADy.

The contents of CPU register zero are undefined after the execution of this analysistion.

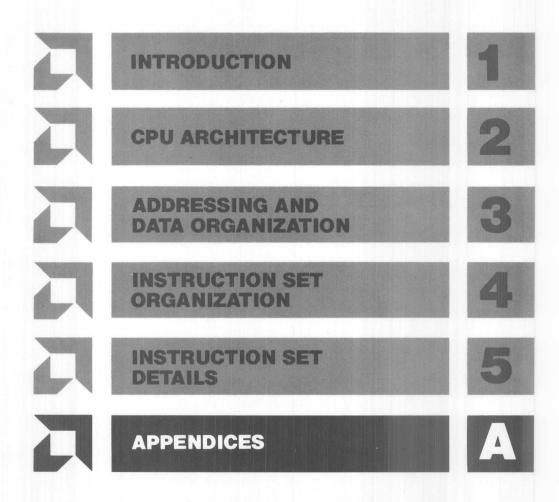
C Z S PW OA H

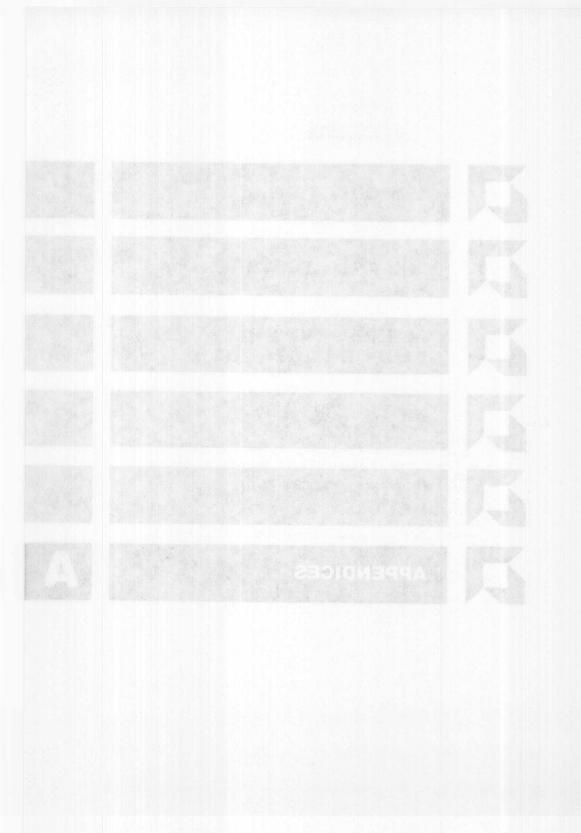
i Masalina i and i

782 =

0 = Clean

Collegates ass - Temosipad = *





APPENDICES

Appendix	Title
Α	AmZ8000 Instruction Set: By Logical Group A-
В	AmZ8000 Instruction Set: Numeric Listing by Opcode B-
C	AmZ8000 Instruction Set: Alphabetic Listing by Mnemonic C-
D	AmZ8000 Instruction Set: Topical Index D-
E	AmZ8000 Instruction Set: Opcode Map E-
F	Executive Module Sample Code F-
G	ASCII Character Set
Н	Powers of 2 and 16 H-
1	Hexadecimal and Decimal Integer Conversion Table



APPENDICES

ACROSCOD: Version 2.0 9/05/80

PROGRAM INSTRUCTIONS;

TEE AMEROOR INSTRUCTION SET';

This program assembles the full set of AmZ8882 nonsegmented instructions, waing each possible opcode and addressing mole combination, More than 100 combinations

The possible addressing modes for a quient

The second second second

APPENDIX A AmZ8000 INSTRUCTION SET: By Logical Group (Alphabetic Within Each Group)

CLEAR, EXCHANGE AND LOAD	A- 2
ARITHMETIC	
LOGICAL ROTATE AND SHIFT	A-10
ROTATE AND SHIFT	A-12
BIT MANIPULATION	A-14
COMPARE	A-16
TRANSLATE	A-18
INPUT/OUTPUT	A-19
PROGRAM CONTROL	
CPU CONTROL	A-24

BASE ADDRESH

GEX INDEXED

PART ADDITES

(PR) PORT REGISTER

ORIGIN #4300;

Short wants from Board Holler was a family barry

NSTRUCT FORES:

: Thata

A

MACRO8000: MACZ B:CODES L F Amz8002 INSTRUCTI		80	Page 1
0000			
0000	PROGRAM INSTRU	CTIONS;	
0000	8		
0000	TITLE 'AmZ80	02 INSTRUCTION SET';	
0000	8		
0000	% This program	assembles the full se	t of
0000	% AmZ8002 nons	egmented instructions,	using
0000	% each possibl	e opcode and addressin	g mode
0000	% combination.	More tham 400 combina	tions
0000	% exist.		
0000	8		
0000		addressing modes for	
0000		are shown in order and	
0000		alues for the purpose	of these
0000	% examples.		
0000	8		
0000	% THE VALUES A	RE:	
0000	8 A 2000	494	
0000	% (IM)	IMMEDIATE	5
0000	e mioro de la valett	N programm	200
0000	% (R)	REGISTER	RH4,
0000	8		R4,
0000	9.		RR4
0000	% (IR)	INDIRECT REGISTER	R2^
0000	9	THE RESIDENCE	NA STATOR COL
0000	% (DA)	DIRECT ADDRESS	LAB
0000	8		
0000	% (RA)	RELATIVE ADDRESS	LAB2
0000	8		
0000	% (X)	INDEXED	LAB(R1)
0000	8		
0000	% (BA)	BASE ADDRESS	R2^(20)
0000	8		
0000	% (BX)	BASE INDEXED	R2^(R1)
0000	8		
0000	% (PA)	PORT ADDRESS	#OFCO
0000		DODE DECICEED	D12
0000	% (PR)	PORT REGISTER	R13
0000			
0000	% ORIGIN	#4300;	
4300	PAGE	51;	
4300	8	51,	
4300	% LAB:	Defined for (DA) and	(X) operand
4300	96	collined tot (DA) and	(v) obergue
1300	INSTRUCTIONS:		
1300	8		
1300	EJECT;		

```
Version 2.0 9/05/80 Page 2

AND LOAD
MACZ B:CODES L P
CLEAR, EXCHANGE, AND LOAD
```

```
MACRO8000: Version 2.0 9/05/80 5 70 Page 3
MACZ B:CODES L P
CLEAR, EXCHANGE, AND LOAD
```

\$ LDRB RH6, LAB; \$ (RA) MOI STAND RELATIVE TO MEMORY \$ LOAD RELATIVE TO MEMORY \$ LOAD RELATIVE TO MEMORY \$ LOAD ADDRESS RELATIVE \$ LOAD MULTIPLE TO REGISTER \$ LOAD MULTIPLE TO MEMORY \$ LOAD MULTIPLE TO ME	CZ	B:CODE	ES L I				Version 2.60			
## LOAD RELATIVE TO REGISTER ## LOAD RELATIVE TO MEMORY ## LOAD ADDRESS RELATIVE ## ## LOAD ADDRESS RELATIVE ## LOAD MULTIPLE TO REGISTER ## LOAD MULTIPLE TO MEMORY ## LOAD AND DECREMENT ## LOAD AND DECREMENT ## LOAD AND DECREMENT ## LOAD AND DECREMENT ## LOAD REA^RA^*, R2^*, R9; ## (IR) MOI ## LOAD AND DECREMENT ## LOAD AND DECREMENT ## LOAD AND INCREMENT ## LOAD AND INC	FΛ				9.					
SO 3006 FF1C LDRB RH6, LAB; RA MOI	E 0					RELATIVE	TO REGISTER			
LDR R6, LAB; (RA) MOI SC LDR R6, LAB; (RA) MOI SC R6 LDR LAB, R6; (RA) MOI SC R6 LDR LAB, R6; (RA) MOI LDR R1, ^LAB; (RA) MOI LDR R1, ^LAB; (RA) MOI LDR R8, LAB, 6; (RA) MOI LDR R8, R8, R2, R9; (RA) MOI LDR R8, R2, R2, R3; (RA) MOI LDR R8,		3006	FFIC		8	IDRR	PH6 IAB.	9	(DA)	MOD
LDRL RR6, LAB; % (RA) MOI LDRL RR6, LAB; % (RA) MOI LDRL RR6, LAB; % (RA) MOI LDRL LAB, RR6; % (IR) MOI LDRR RR6, R2^, R9; % (IR) MOI LDRR R8^, R2^, R9; % (IR) MOI LDRR										
## LOAD RELATIVE TO MEMORY ## LOAD LOAD RELATIVE ## LOAD MULTIPLE TO REGISTER ## LOAD MULTIPLE TO REGISTER ## LOAD MULTIPLE TO MEMORY ## LOAD MOLTIPLE TO MEMORY ## LOAD MOLTPLE										
## LOAD RELATIVE TO MEMORY ## LOAD LAB, RH6;	EC				ફ				(/	
## LDRB LAB,RH6;										
LOAD MULTIPLE TO MEMORY LOAD MODERMENT LOAD MAD DECREMENT LOAD MODERMENT LOAD MAD DECREMENT LOAD MODERMENT LOAD MAD DECREMENT LOAD MODERMENT LOAD MAD MODERMENT LOAD MODERMENT, AND REPEAT LOAD MODERMENT, AND REPEAT LOAD MAD INCREMENT LOAD MODERMENT LOAD MAD INCREMENT LOAD MA						RELATIVE	TO MEMORY			
### LOAD MULTIPLE TO MEMORY ### LOAD AND DECREMENT ### LOAD AND DECREMENT, AND REPEAT ### LOAD, DECREMENT, AND REPEAT ### LOAD AND INCREMENT ###			الماليات		8					
LDRL LAB,RR6; % (RA) MOI LORD ADDRESS RELATIVE LDR R11, LAB; % (RA) MOI REC										
*** *** *** *** *** *** *** *** *** **										
\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$		3/06	60,44		0.	LDRL	LAB, RR6;	8	(RA)	MOD
\$ LOAD ADDRESS RELATIVE \$ 100										
8	-					ADDRESS R	ELATIVE			
## LOAD MULTIPLE TO REGISTER ## LOAD MULTIPLE TO MEMORY ## LOAD AND DECREMENT ## LOAD AND DECREMENT ## LOAD AND DECREMENT ## LOAD AND DECREMENT ## LOAD R8^,R2^,R9; ## (IR) MOI ## LOAD, DECREMENT, AND REPEAT ## LOAD, DECREMENT ## LOAD AND INCREMENT	8									
## LOAD MULTIPLE TO REGISTER ## LOAD MOUNT R8, LAB, 6;	8	340B	FF04			LDR	R11, LAB;	ક	(RA)	MOD
## LOAD MULTIPLE TO REGISTER ## LOAD MULTIPLE TO MEMORY ## LOAD AND LAB, R8, 6; ## (IR) MOI ## LOAD AND DECREMENT ## LOAD AND DECREMENT ## LOAD AND DECREMENT ## LOAD R8^, R2^, R9; ## (IR) MOI ## LOAD, DECREMENT, AND REPEAT ## LOAD, DECREMENT, AND REPEAT ## LOAD, DECREMENT, AND REPEAT ## LOAD AND INCREMENT ##	rC				8				,	
## CC 1C21 0805 LDM R8,R2^,6; % (IR) MOI SC11 0805 4300 LDM R8,LAB,6; % (DA) MOI SC11 0805 4300 LDM R8,LAB,6; % (X) MOI SC11 0805 4300 LDM R8,LAB,6; % (IR) MOI SC10 0805 LDM R2^,R8,6; % (IR) MOI SC19 0805 4300 LDM LAB,R8,6; % (DA) MOI LDM LAB,R8,6; % (DA) MOI LDM LAB,R8,6; % (X) MOI LDM LAB,R8,6; % (X) MOI LDM LAB,R8,6; % (IR) MOI LDM LAB,R8,6; % (IR) MOI LDM R8^,R2^,R9; % (IR) MOI LDM R8^,R2^,R2^,R9; % (IR) MOI LDM LDM R8^,R2^,R2^,R9; % (IR) MOI LDM LDM R8^,R2^,R2^,R9; % (IR) MOI LDM L										
LDM R8,R2^,6; % (IR) MOI 500 501 0805 4300 LDM R8,LAB,6; % (DA) MOI 501 0805 4300 LDM R8,LAB(R1),6; % (X) MOI 60 501 0805 4300 LDM R8,LAB(R1),6; % (X) MOI 61 00						MULTIPLE	TO REGISTER			
SCO1 0805 4300 LDM R8, LAB, 6; % (DA) MOI		1001	0005		8		DO DOO 6			
LDM R8, LAB(R1), 6; % (X) MOI R8, LAB(R1), 6; % (X) MOI R8, LAB(R1), 6; % (X) MOI R0C R LDM R2^, R8, 6; % (IR) MOI LDM LAB, R8, 6; % (IR) MOI RC R LDDB R8^, R2^, R9; % (IR) MOI LDD R8^, R2^, R9; % (IR) MOI LDDR R8^, R2^, R9; % (IR				1200						
% LOAD MULTIPLE TO MEMORY % LOAD MULTIPLE TO MEMORY % LOM R2^,R8,6; % (IR) MOI 5C09 0805 4300 LDM LAB,R8,6; % (DA) MOI 6C 5C19 0805 4300 LDM LAB(R1),R8,6; % (X) MOI 6C % 6C % LOAD AND DECREMENT % 6C BA29 0988 LDDB R8^,R2^,R9; % (IR) MOI 6A4 % LOAD, DECREMENT, AND REPEAT % LOAD, DECREMENT, AND REPEAT % LOAD, DECREMENT, AND REPEAT % LOAD R8^,R2^,R9; % (IR) MOI 6A4 % LOAD, DECREMENT, AND REPEAT % LOAD R8^,R2^,R9; % (IR) MOI 6B5 BB29 0980 LDDR R8^,R2^,R9; % (IR) MOI 6B6 BB29 0980 LDDR R8^,R2^,R9; % (IR) MOI 6B6 BB29 0980 LDDR R8^,R2^,R9; % (IR) MOI 6B6 BB29 0980 LOAD AND INCREMENT % LOAD AND INCREMENT % LOAD AND INCREMENT % LOAD R8^,R2^,R9; % (IR) MOI 6B6 BA21 0988 LDIB R8^,R2^,R9; % (IR) MOI							,			
\$ LOAD MULTIPLE TO MEMORY \$ LOAD MULTIPLE TO MEMORY \$ LOAD MULTIPLE TO MEMORY \$ LOAD R2^,R8,6; \$ (IR) MOI \$ LOM LAB,R8,6; \$ (DA) MOI \$ LOM LAB(R1),R8,6; \$ (X) MOI \$ LOAD AND DECREMENT \$ LOAD AND DECREMENT \$ LOAD R8^,R2^,R9; \$ (IR) MOI \$ LOAD R8^,R2^,R9; \$ (IR) MOI \$ LOAD, DECREMENT, AND REPEAT \$ LOAD, DECREMENT, AND REPEAT \$ LOAD R8^,R2^,R9; \$ (IR) MOI \$ LO		3011	0003	4300	9	LDM	RO, LAB (R1), 0;	8	(X)	MOD
\$ LOAD MULTIPLE TO MEMORY \$ LOAD MULTIPLE TO MEMORY \$ COC										
\$ LDM R2^,R8,6; \$ (IR) MOI SC09 0805 4300 LDM LAB,R8,6; \$ (DA) MOI LDM LAB(R1),R8,6; \$ (X) MOI LDM R8^,R2^,R9; \$ (IR) MOI LDD R8^,R2^,R9; \$ (IR) MOI LDD R8^,R2^,R9; \$ (IR) MOI LDDR R8^,R2^,R2^,R9; \$ (IR) MOI LDDR R8^,R2^,R2^,R9; \$ (IR) MOI LDDR R8^,R2^,R2^,R9; \$ (IR) MOI LDDR R8^,R					_	MULTIPLE	TO MEMORY			
LO 5C09 0805 4300 LDM LAB,R8,6; % (DA) MOI DECREMENT LC % LOAD AND DECREMENT LC % LOAD R8^,R2^,R9; % (IR) MOI DECREMENT, AND REPEAT LOAD BB29 0980 LDDR R8^,R2^,R9; % (IR) MOI DECREMENT, AND REPEAT LOAD BB29 0980 LDDR R8^,R2^,R9; % (IR) MOI DECREMENT, AND REPEAT LOAD R8^,R2^,R9; % (IR) MOI DECREMENT, AND REPEAT LOAD R8^,R2^,R9; % (IR) MOI DECREMENT										
LOM LAB,R8,6; % (DA) MOI LOM LAB(R1),R8,6; % (X) MOI LOM LAB,R8,6; % (DA) MOI LOM LAB,R8,6; % (DA) MOI LOM LAB,R8,6; % (DA) MOI LOM LAB,R8,6; % (X) MOI LOM LAB,R8,6; LOD RAP,REMENT LOD RAP,	C	1C29	0805			LDM	R2^,R8,6;	8	(IR)	MOD
LDM LAB(R1),R8,6; % (X) MOI LC	LO					LDM		96	(DA)	MOD
\$ LOAD AND DECREMENT \$ LOAD AND DECREMENT \$ LOBB R8^,R2^,R9;	16	5C19	0805	4300		LDM	LAB(R1), R8, 6;			MOD
\$ LOAD AND DECREMENT \$ LOAD AND DECREMENT \$ LOBB R8^,R2^,R9;										
\$ LDDB R8^,R2^,R9; \$ (IR) MOI BB29 0988 LDDD R8^,R2^,R9; \$ (IR) MOI LDD R8^,R2^,R9; \$ (IR) MOI LDDR R8^,R2^,R9; \$										
LDDB R8^,R2^,R9; % (IR) MOI BB29 0988 LDDD R8^,R2^,R9; % (IR) MOI LDD R8^,R2^,R9; % (IR) MOI LDDR R8^,R2^,R9; % (IR) MOI						AND DECR	EMENT			
BB29 0988 LDD R8^,R2^,R9; % (IR) MOI LDD R8^,R2^,R9; % (IR) MOI LDDR R8^,R2^,R9; % (IR) MOI		DAZO	0000		ð	LDDD	DO^ DO^ DO	0	(TD)	MOD
% LOAD, DECREMENT, AND REPEAT % LOAD, DECREMENT, AND REPEAT % LOAD R8^,R2^,R9; % (IR) MOI LDDR R8^,R2^,R9; % (IR) MOI CC % LOAD AND INCREMENT % LOAD AND INCREMENT % LOAD R8^,R2^,R9; % (IR) MOI CC & % LOAD AND R8^,R2^,R9; % (IR) MOI CC & % LOAD R8^,R2^,R2^,R9; % (IR) MOI CC &										
% LOAD, DECREMENT, AND REPEAT % LOAD, DECREMENT, AND REPEAT % LOAD R8^,R2^,R9; % (IR) MOI LDDR R8^,R2^,R9; % (IR) MOI CC % LOAD AND INCREMENT % LOAD AND INCREMENT % LOAD R8^,R2^,R9; % (IR) MOI % LDIB R8^,R2^,R9; % (IR) MOI		0025	0,000		8	LDD	NO , NZ , NS;	ъ	(IK)	MOD
% LDDRB R8^,R2^,R9; % (IR) MOI LDDR R8^,R2^,R9; % (IR) MOI LDDR R8^,R2^,R9; % (IR) MOI RC R8^,R2^,R2^,R9; % (IR) MOI RC R8^,R2^,R2^,R2^,R9; % (IR) MOI RC R8^,R2^,R2^,R2^,R2^,R2^,R2^,R2^,R2^,R2^,R2										
8 24 BA29 0980 LDDRB R8^,R2^,R9; % (IR) MOI 28 BB29 0980 LDDR R8^,R2^,R9; % (IR) MOI 39 CC % LOAD AND INCREMENT 30 CC BA21 0988 LDIB R8^,R2^,R9; % (IR) MOI	24				% LOAD	, DECREME	NT, AND REPEAT			
LDDR R8^,R2^,R9; % (IR) MOI CC % CC % LOAD AND INCREMENT CC % CC BA21 0988 LDIB R8^,R2^,R9; % (IR) MOI	24									
% % % % LOAD AND INCREMENT % (IR) MOI							R8^,R2^,R9;			
% LOAD AND INCREMENT % BA21 0988 LDIB R8^,R2^,R9; % (IR) MOI		BB29	0980		0	LDDR	R8^,R2^,R9;	ક	(IR)	MOD
% LOAD AND INCREMENT % CC 8A21 0988 LDIB R8^,R2^,R9; % (IR) MOI										
% LDIB R8^,R2^,R9; % (IR) MOI						AND THER	PMPNIT			
2C BA21 0988 LDIB R8^,R2^,R9; % (IR) MOI						AND INCR	EN EN I			
		BA21	0988		0	I.DTR	R8^ R2^ RQ.	9	(TP)	MOD

MACRO8000: Version 2.0 8 9/05/80 9 00 00 00 Page Page 5 CLEAR, EXCHANGE, AND LOAD 4434 4434 4434 % LOAD, INCREMENT, AND REPEAT LDIRB R8^,R2^,R9; % (IR) MODE LDIR R8^,R2^,R9; % (IR) MODE 4434 BA21 0980 4438 BB21 0980 443C LORE - RROLL 901 EJECT; 443C

```
MACRO8000: Version 2.0 9/05/80
                                                                  Page 6
MACZ B:CODES L P
STACK MANIPULATION
                             TITLE STACK MANIPULATION';
443C
443C
                             8 POP WITH CARRY TO 8
443C
443C
                              8
                                       POP R4,R12^; % (R) MODE
POP R2^,R12^; % (IR) MODE
POP LAB,R12^; % (DA) MODE
POP LAB(R1),R12^; % (X) MODE
      97C4
443C
      17C2
443E
        57C0 4300
4440
4444
        57C1 4300
4448
                           POPL RR4,R12^; % (R) MODE
POPL R2^,R12^; % (IR) MODE
POPL LAB,R12^; % (DA) MODE
POPL LAB(R1),R12^; % (X) MODE
      95C4
4448
444A 15C2
444C 55C0 4300
4450 55C1 4300
4450 55C1 4300
                                      ADDB
ADDB
4454 gom Ag
4454
4454
8 PUSH
                         R6,5; 8
                                                R12^,5; % (IM) MODE
R12^,R4; % (R) MODE
R12^,R2^; % (IR) MODE
R12^,LAB; % (DA) MODE
R12^,LAB(R1); % (X) MODE
4454
                                       PUSH
        ODC9 0005
4454
                                       PUSH
        93C4
13C2
4458
                                       PUSH
445A
                                      PUSH
        53C0 4300 PUSH
53C1 4300 PUSH
445C
4460
      91C4
11C2
                          RR6,5,
4464
                                                R12^,RR4; % (R) MODE
R12^,R2^; % (IR) MODE
R12^,LAB; % (DA) MODE
R12^,LAB(R1); % (X) MODE
                                       PUSHL
4464
4466
                                       PUSHL
         51C0 4300
4468
                                       PUSHL
446C
         51C1 4300
                                       PUSHL
                      RRE, L&B(R1);
4470
4470
                                       EJECT:
```

```
MACRO8000: Version 2.0 9/05/80 solution
                                                        Page 7
MACZ B:CODES L P
ARITHMETIC
         TITLE 'ARITHMETIC';
4470
                             8
                             % ADD WITH CARRY
4470
4470
                             8
                                      ADCB RH6,RH4;
ADC R6,R4;
4470 B446
44/0 B446
4472 B546
4474
4474
                                                               % (R) MODE
                     R6,R4;

R6,R4;

R6,R4;
                                                                  % (R) MODE
4474
4474
                                      ADDB RH6,5; % (IM) MODE
ADDB RH6,RH4; % (R) MODE
ADDB RH6,R2^; % (IR) MODE
ADDB RH6,LAB; % (DA) MODE
ADDB RH6,LAB(R1); % (X) MODE
4474 0006 0505
        8046
0026
                                      ADDB
447A
447C
        4006 4300
4480
        4016 4300
4484
                                               R6,5; % (IM) MODE
R6,R4; % (R) MODE
R6,R2^; % (IR) MODE
R6,LAB; % (DA) MODE
R6,LAB(R1); % (X) MODE
        0106 0005 ADD
8146 ADD
0126 ADD
4106 4300 ADD
4116 4300 ADD
4484
4488
448A
448C
4490
4494
                                       ADDL RR6,5; % (IM) MODE ADDL RR6,RR4; % (R) MODE ADDL RR6,R2^; % (IR) MODE ADDL RR6,LAB; % (DA) MODE ADDL RR6,LAB(R1); % (X) MODE
        1606 0000 0005
9646
1626
4494
449A
449C
449E
        5606 4300
        5616 4300
44A2
44A6
44A6
44A6
                              % DECIMAL ADJUST BYTE
44A6
                              8
44A6 B040
                                       DAB RH4;
                                                             % (R) MODE
44A8
                              8
44A8
                              8
44A8
                             % DECREMENT
44A8
                             8
                                              RH4,12; % (R) MODE
R2^,12; % (IR) MODE
LAB,12; % (DA) MODE
LAB(R1),12; % (X) MODE
44A8 AA4B
                                       DECB
44AA 2A2B
                                       DECB
44AC 6A0B 4300
                                     DECB
44B0 6A1B 4300
                                       DECB
44B4
                             90
                                              R4,12; % (R) MODE
R2^,12; % (IR) MODE
LAB,12; % (DA) MODE
LAB(R1),12; % (X) MODE
44B4
      AB4B
                                       DEC
44B6
       2B2B
                                       DEC
44B8
       6B0B 4300
                                       DEC
44BC
        6B1B 4300
                                       DEC
44C0
44C0
                                      EJECT;
```

```
MACRO8000: Version 2.0 9/05/80 0.5 401818V
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       Page 8
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        MACK B: CODES L P
            MACZ B:CODES L P
### ARITHMETIC

### ARITHMETIC
             ARITHMETIC
```

```
Version 2.0 08 9/05/80 0.5 noisiev
             MACR08000:
             MACZ B: CODES L P
             ARITHMETIC
4566 1226

4568 5206 4300

456C 5216 4300

4570 dow (A) $ SUBL RR6, LAB;

RR6, LAB (R1);

** SUBL RR6,
```

MACZ B:CODES L P LOGICAL						
4570 4570 8004 (8) 8	TITLE		L';			
4570 BUOM (RI) P	% AND					
4570 0607 0505	LAB; 8 LAB(RI);				(TM)	MODE
4574 8647		ANDB	RH7, RH4;		(R)	
4576 0627		ANDB	RH7, R2^;	%	(IR)	MODE
4578 4607 4300			RH7, LAB;	ક	(DA)	MODE
457C 4617 4300			RH7, LAB(R1);	8	(X)	MODE
4580 300M (X) #	\$ LA B (A L &					
4580 0707 0005		AND	R7,5;			MODE
4584 8747		AND	R7,R4;		(R)	
4586 0727			R7,R2^;			MODE
4588 4707 4300 458C 4717 4300		AND	R7, LAB; R7, LAB(R1);			MODE
4590	8 4 9 4 9			8		MODE
4590	96					
4590	% COMPL	EMENT				
4590	%	IVE OR				
4590 8C40		COMB	RH4;	96	(R)	MODE
4592 OC20		COMB	R2^;			MODE
4594 4C00 4300				8	(DA)	MODE
4598 4C10 4300		COMB	LAB(R1);		(X)	
459C (49)	RHY, LAS	BROX				
459C 8D40						MODE
459E 0D20 45A0 4D00 4300		COM	R2^;			MODE
45A4 4D10 4300		COM	LAB; LAB(R1);			MODE MODE
45A8 45A8	8 28 78	ROX		6		60 909
45A8 TOM (AC)	HY, LAES					
45A8 DOM (X)	% OR (9)					
45A8	8		8			
45A8 0407 0505		ORB	RH7,5;			MODE
45AC 8447		ORB	RH7,RH4;		(R)	
45AE 0427		ORB	RH7,R2^;			MODE
45B0 4407 4300		ORB	RH7, LAB;			MODE
45B4 4417 4300		ORB	RH7, LAB(R1);	8	(X)	MODE
45B8 0507 0005	8	OD	חק ר	0	(T M)	морп
45B8 0507 0005 45BC 8547		OR OR	R7,5; R7,R4;		(IM) (R)	MODE
45BE 0527		OR	R7,R4;			MODE MODE
45C0 4507 4300		OR	R7, LAB;			MODE
45C4 4517 4300		OR	R7, LAB(R1);		(X)	MODE
45C8	8					
45C8	8					
45C8	% TEST					
45C8	8					
45C8 8C44		TESTB	RH4;	8		MODE
45CA 0C24		TESTB	R2^;		(IR)	
45CC 4C04 4300		TESTB	LAB;	8	(DA)	MODE

```
MACR08000:
                                                                                                                                                                                     Version 2.0 9/05/80 0.5 molerey Page 11
        MACZ B: CODES L P
           LOGICAL
45D4
45D4
8D44
45D6
0D24
5D8
4D04
4300
TEST
CAB;
$ (IR) MODE
45D8
4D04
4300
TEST
LAB;
$ (DA) MODE
45D0
45E0

45E0
9C48
5E2
45E2
45E2
5C08
4300
TESTL
CAB(R1);
$ (X) MODE
45E2
45E4
5C08
4300
TESTL
CAB(R1);
$ (IR) MODE
45EC
45EC
5 (IR) MODE
45EC
45EC
6 (IR) MODE
45EC
45EC
6 (IR) MODE
45EC
6 (IR) MODE
45EC
6 (IR) MODE
7 
                                             ORB R7.5; % (IM) MODE
OR R7.64; % (R) MODE
OR R7.82'; % (IR) MODE
OR R7.LAB; % (DA) MODE
OR R7.LAB; % (X) MODE
```

MACZ	B:CODES	LP				Version 2.0 0			00:8	
ROTAT	E AND SHI	FT								
4610			TI	TLE	'ROTATE	AND SHIFT';				
4610										
4610						IGIT				
			8						824	
4610	BE47				RLDB A	RH7,RH4;	8	(R)	MODE	
4612			ક							
4612						DIGIT				
4612	D G 4 F		8		DOU TERM	THIEF				
	BC47				RRDB	RH7, RH4;	ક	(R)	MODE	
			8							
					E LEFT					
4614	B240		9		DID	DIIA 1.	0	(D)	MODE	
	B340				KLB	RH4,1;	*	(R)	MODE MODE	
4618	B340				RL	R4,1;	8	(R)	MODE	
			90							
4610			5	ROTATI	SRAL	HROUGH CARRY				
	B248		6		DICB	RH4,1;	Q	(D)	MODE	
	B348				RIC	R4;	9	(R)	MODE	
461C	-510		96		od Thorn	TTIHE &	0	(14)		
461C			98							
461C			8	ROTATI	E RIGHT					
461C			96							
461C	B244				RRB	RH4,1;	8	(R)	MODE	
	B344				RR	R4,1; 8	%	(R)	MODE	
			8							
4620			8							
4620				ROTATI	E RIGHT	THROUGH CARRY				
4620			ક							
	B24C				RRCB	RH4,1;			MODE	
4622	B34C		0		RRC	R4,1;	જ	(R)	MODE	
4624			96							
4624			90	CHIDM	DVNIAMTO	ADIMIMPMIC				
4624			90	SHIFT	DYNAMIC	ARITHMETIC				
	B24B 09	200	8		SDAR	RH4,R9;	Q	(P)	MODE	
4628	B34B 09	900			SDA	R4, R9;		(R)		
	B34F 09					RR4, R9;			MODE	
4630	D341 03	, 00	96		DDAL	MA, MJ,	6	(14)	HODE	
4630			96							
4630			%	SHIFT	DYNAMIC	LOGICAL				
4630			8							
4630	B243 09	900			SDLB	RH4, R9;	0,0	(R)	MODE	
4634	B343 09	900			SDL	R4, R9;	96	. ,	MODE	
4638	B347 09	900			SDLL	RR4, R9;		(R)	MODE	
463C			96							
463C					EJECT;					

MACZ B:CODES L P ROTATE AND SHIFT	Version 2.0 9/05/80 0.5 noime	Page 13
463C	% SHIFT LEFT ARITHMETIC % SLAB RH4,2; SLA R4,2; SLAL RR4,2; % % SHIFT LEFT LOGICAL %	
463C	\$ SHIFT LEFT ARITHMETIC	
463C	% TIDIO TIDIO ATATOR A	
463C 463C B249 0002 4640 B349 0002	SLAB RH4.2:	% (R) MODE
4640 B349 0002	AMS THE SLA SOLE R4.2:	% (R) MODE
4644 B34D 0002	SLAL RR4,2;	% (R) MODE
4648	8	612
4648	% TIDIO THOIR STATES &	
4648	% SHIFT LEFT LOGICAL	
4648	& HALLHAM SQRR	
4648 B241 0002 464C B341 0002	DLLD KII4, Z;	6 (K) PIUDE
464C B341 0002	SLL R4,2;	% (R) MODE
4650 B345 0002	SLL R4,2; SLLL RR4,2;	% (R) MODE
4654 4654	8	
4654	% SHIFT RIGHT ARITHMETIC	
1654	& SHIFT RIGHT ARTTHMETIC	
4654 R249 FFFF	SRAB RH4,2; SRA R4,2; SRAL RR4,2;	& (D) MODE
4658 B349 FFFE	SRA R4.2.	& (R) MODE
465C B34D FFFE	SRAI. RR4.2:	% (R) MODE
4660 GGGM (8)	8 . 1 . 1 . 1 . 1	8ASA RIA
4660 3004 (8) 8	8	
4660	% SHIFT RIGHT LOGICAL	
4660	8	
4660 B241 FFFE	SRLB RH4,2;	% (R) MODE
4664 B341 FFFE	SRL R4,2;	% (R) MODE
4668 B345 FFE	SRLL RR4,2;	% (R) MODE
466C	* EJECT;	
	SRA R4,2; SRAL RR4,2; SRAL RR4,2; SRLB RH4,2; SRL R4,2; SRL R4,2; SRLL RR4,2; SRLT; BEJECT;	

				1 2.0	9/05/	Version 2.0 08			Page	e 1
	B:CODE ANIPULA	ES L P							B:CC	
		111011								
466C			Т	ITLE	'BIT M	ANIPULATION';				
466C			ક							
466C			ક	TEST	BIT STA	TIC TEE #				
466C			ક							
466C	A640					RH4,0;				
	2620				ABITB T	R2^,0;			MODE	
	6600				BITB					
	6610	4300	1911		BITB	LAB(R1),0;	8	(X)	MODE	
4678			8			WA TEST AND				
	A740				BIT	R4,0;			MODE	
467A	2720	用) 8				R2^,0;				
	6700					LAB,0;	8	(DA)	MODE	
	6710								MODE	
					U STE					
4684			8		DIM DIII					
					BIT DYN					
	2606		96						000	
	2606 2706					RH4, R6;			MODE	
	2/06	0400			BIT	R4,R6;	8	(R)	MODE	
468C 468C			8							
468C					r BIT ST					
468C			9		I BII SI	ATIC				
	A 2 4 0		6		RESB	RH4,0;	Q	(D)	MODE	
468E	A240 2220				RESB	R2^,0;			MODE	
	6200				RESB			-	MODE	
	6210				RESB			(X)		
4698			96			2.12 (1.12), 0,	0	(11)		
4698	A340				RES	R4,0;	8	(R)	MODE	
469A	2320				RES	R2^,0;			MODE	
469C	6300	4300			RES	LAB, 0;			MODE	
46A0	6310	4300			RES	LAB(R1),0;		(X)		
46A4			8							
46A4			96							
46A4				RESET	r BIT DY	NAMIC				
46A4			8							
	2206					RH4, R6;		(R)		
	2306	0400			RES	R4, R6;	8	(R)	MODE	
46AC			96							
46AC			8							
46AC				SET E	BIT STAT	IC				
46AC 46AC	A440		8		CEMP	DIIA O.	0	(D)	MODE	
46AE	2420				SETB SETB	RH4,0;	8	(R)	MODE	
46B0		4300			SETB	R2^,0; LAB,0;	8	(IR)		
46B4		4300			SETB	LAB,0; LAB(R1),0;	90 90	(DA)	MODE	
46B8	0410	4500	8		DEID	PWD (KI) 'O!	6	(X)	MODE	
46B8	A540		8		SET	R4,0;	010	(R)	MODE	
46BA	2520				SET	R4,0; R2^,0;	96	(R)	MODE	
46BC		4300			SET	LAB, 0;	90	(DA)		
46C0		4300			SET	LAB(R1),0;	96	(X)	MODE	
	0010	1000			DIL	DUD (ILT) 101	0	(V)	HODE	

MACRO8000: MACZ B:CODES L P BIT MANIPULATION	Version 2.0 09/05/80 0.8 notageV	Page 15
16C4 16C4 16C4 16C4 16C4 16C4 16C8 16C8 16CC	% TEST AND SET % TSETB RH4; TSETB R2^; TSETB LAB; TSETB LAB(R1); % TSET R4; TSET R2^; TSET LAB;	% (R) MODE % (IR) MODE % (DA) MODE % (X) MODE
16E4	EJECT;	

```
MACRO8000: Version 2.0 9/05/80
                                                                    Page 16
MACZ B: CODES L P
COMPARE
46E4
                             TITLE
                                       'COMPARE';
46E4
                 NT. SID REPEAT
46E4
                             % COMPARE REGISTER WITH MEMORY
46E4 0A06 0505 CPB RH6,5;
                                                RH6,5; % (IM) MODE
RH6,RH4; % (R) MODE
RH6,R2^; % (IR) MODE
RH6,LAB; % (DA) MODE
RH6,LAB(R1); % (X) MODE
         8A46
                                       CPB
46EA
         0A26
                                       CPB
        4А06 4300 мамаярая дил СРВ
46EC
46F0
        4A16 4300

0B06 0005

CP R6,R4; % (R) MODE

0B26

CP R6,R2^; % (IR) MODE

4B06 4300

CP R6,LAB; % (DA) MODE

AB16 4300

CP R6,LAB(R1); % (X) MODE
         4A16 4300
                                       CPB
46F4
46F4
46F8
46FA
46FC
        4B16 4300

1006 0000 0005

CPL RR6,5;

9046

CPL RR6,RR4;

1026

CPL RR6,R2^;

CPL RR6,LAB;

RR6,LAB;

RR6,LAB(R1);
4700
4704
                                                                 % (IM) MODE
% (R) MODE
% (IR) MODE
4704
470A
470C
                                                                  % (DA) MODE
470E
4712
                                                 RR6, LAB(R1); % (X) MODE
4716

      0C21 0505
      CPB
      R2^,5;
      % (IR) MODE

      4C01 4300 0505
      CPB
      LAB,5;
      % (DA) MODE

      4C11 4300 0505
      CPB
      LAB(R1),5;
      % (X) MODE

4716
471A
4720
         0D21 0005
4726
                             CP SIEGO
                                       CP R2^,5; % (IR) MODE
CP LAB,5; % (DA) MODE
CP LAB(R1),5; % (X) MODE
4726
472A
         4D01 4300 0005
4730
         4D11 4300 0005
4736
4736
4736
                              % COMPARE AND DECREMENT
4736
4736
        BA28 0765
                                       CPDB
                                                 RH6, R2, R7, MI; % (IR) MODE
473A
         BB28 0765
                                       CPD
                                                 R6, R2, R7, MI; % (IR) MODE
473E
                              8
473E
                              8
473E
                              % COMPARE, DECREMENT, AND REPEAT
473E
473E
         BA2C 0765
                                       CPDRB
                                                 RH6, R2^, R7, MI; % (IR) MODE
4742
         BB2C 0765
                                       CPDR
                                                 R6, R2^, R7, MI; % (IR) MODE
4746
                              8
4746
                              0
4746
                              % COMPARE AND INCREMENT
4746
4746
         BA20 0765
                                       CPIB
                                                 RH6, R2^, R7, MI; % (IR) MODE
                                                 R6, R2^, R7, MI; % (IR) MODE
474A
         BB20 0765
                                       CPI
474E
```

COMPA	B: C	: ODES			n 2.0	9/05/80	2-0 (Pag	
474E				9.		COMPARI						
474E 474E					COMPARI	E, INCRE	EMENT, A	AND REPEA	Т			
474E		24 07			(CPTRB	RH6.R2	,R7,MI;	ક (IR)	MODE	
4752 4756	ВВ	24 07		9	RH6, RH4	CPIR 890	R6,R2^	R7,MI;	8	(IR)	MODE	
4756				8		690						
4/50				8	COMPARI	E STRING	G AND DE	ECREMENT				
4756 4756		2A 07		*		CPSDB	R11^.R	2^,R7,NE;				
475A		2A 07			R6, 57		R11 , R2	2^,R7,NE;	8	(IR)	MODE	
475E				8								
475E 475E				8		E CTDING	DEC	AND REPE	יתר			
475E				9	COMPARI	E SIKING	o, DEC.	AND REPE	300			
475E	ВА	2E 07	7BE					2^,R7,NE;				
4762	ВВ	2E 07	7BE		RRE,5;	CPSDR	R11^, R2	2^,R7,NE;	8	(IR)	MODE	
4766 4766				8								
4766				8	COMPAR	E STRING	G AND IN	NCREMENT				
4766		(X)		8	CONTINIC	743				16 4		
4766 476A		22 07				CPSIB		2^,R7,NE; 2^,R7,NE;				
476E	DD	22 0	706	BTAIGEMM	BTIW Y	HOMEN B	COMPAR	, KI, NE,	0	(111)	HODE	
476E				8	.27532				202			
476E 476E				8	COMPAR	E STRING	J, INC.	AND REPE	AT			
476E 4772	BA BB	26 0				CPSIRB CPSIR		2^,R7,NE; 2^,R7,NE;				
		26 U	/ DE				/	c , IN , INL,	8	()		
4//0						EJECT;						
4776						EJECT;						
4776												
4776												
4776				,5; r ,87,m1;								
4776												
4776				,5; r ,87,m1;								
4776				,5; r ,87,m1;								
4776				, 5; ra, mi; ea, mi; nno reper		CPDE CPDE CPDE CPD						
4776				1,5; 1 1,87,MI; 1,87,MI;								
4776				.5; .R7,MI; .E7,MI; AND REPEA		CPDE CPDE CPDE CPD CPD CPD CPD CPDR						
4776				.5; .R7,M1; .R7,M1; AND REPEA .R7,M1;		CPDB CPDB CPD CPD CPDR CPDR CPDR						
4776				.5; .R7,M1; .R7,M1; AND REPEA .R7,M1;		CPDE CPDE CPDE CPD CPD CPD CPD CPDR						
4776				.5; .R7,M1; .R7,M1; AND REPEA .R7,M1;		CPDB CPDB CPD CPD CPDR CPDR CPDR						

	B:CODES L P	Version 2.0 09/05/80 0.5 noisiay Pa	
4776		TITLE 'TRANSLATE';	
4776		Q IRANDLAIE ;	
4776		% TRANSLATE AND DECREMENT	
4776		8	
4776	B8B8 0620	TRDB gar Rll^,R2^,R6; % (IR) MOD	E
477A		BAAA OFCO THE A RHA, ROLSON	
477A		8	
477A		% TRANSLATE, DECREMENT, AND REPEAT	
477A	COM (AS) 8	TRDRB Rll^,R2^,R6; % (IR) MOD	
	B8BC 0620	TRDRB R11, R2, R6; % (IR) MOD	E SA
477E 477E		* THE STATE OF THE	
477E		% TRANSLATE AND INCREMENT	
477E	B8B0 0620	TRIB R11^,R2^,R6; % (IR) MOD	E SA
4782		8	
4782		8	
4782		* TRANSLATE, INCREMENT AND REPEAT	
4782		8	
	B8B4 0620	TRIRB R11^,R2^,R6; % (IR) MOD	E
4786		38D8 0920 R2", R13 R9;	
4786		\$ TDANCIATE AND TECH DECDEMENT	
4786		% TRANSLATE AND TEST, DECREMENT	
4786	B8BA 0620	TRTDB Rll^.R2^.R6: % (IR) MOD	E 00
478A		3ADO 0928 INI 8 827, R1 8 R9; INI 3ADO 0928 INI 3ADO 0928	SB
478A		38DD 0928 INI NI R2 R1 8 R9; TRI	
478A		% TRANSLATE AND TEST, DEC. AND REPEAT	
478A		8	
	B8BE 062E	TARGER CHA THE TRTDRB R11, R2, R6; % (IR) MOD	E Aas
478E		8	
4/8E		% TRANSLATE AND TEST, INCREMENT	
478E 478E		* TRANSLATE AND TEST, INCREMENT 00 0082	
	B8B2 0620	TRTIB R11^,R2^,R6; % (IR) MOD	F
4792	3052 0020	RII , RZ , RO; & (IR) MOD	201
4792		8	
		% TRANSLATE AND TEST, INC. AND REPEAT	
4792		8740 00704 8740 00700 8144	
4792	B8B6 062E	TRTIRB R11^,R2^,R6; % (IR) MOD % 18,819	E 85
4796		SED4 RI3, RI3, RI3, RI	
4/96		AN, 00004 EJECT;	

4796 4796 4796 4796 4796 4796 4798 3ACD4 4798 3A44 0FC0 ** INB RH4,#0FC0; ** (PR) MODE 479C 479C 3DD4 IN R4,R13; ** (PR) MODE 479E 3B44 0FC0 IN R4,#0FC0; ** (PA) MODE 47A2 47A2 47A2 47A2 47A2 3AD8 0928 INDB R2^,R13,R9; ** (IR,PR) MODE 47AA 47AA 47AA 47AA 47AA 47AA 47AA 47A	MACRO8 MACZ INPUT/	B:CODE	ES L P		2.0	9/05/8	Version 2.0 08			age 19
4796 4796 4796 4796 4798 3A44 0FC0 INB RH4,#0FC0; % (PA) MODE 479C 3DD4 479C 3DD4 479C 3DD4 479C 3DD4 47A2 47A2 47A2 47A2 47A2 47A2 47A4 47AA 47AA	4796			Т	ITLE	'INPUT	OUTPUT!;			
4796 4796 4796 4796 3CD4 4798 3A44 OFCO 1NB RH4,#0FCO; % (PA) MODE 4792 4792 3DD4 4798 3B44 OFCO 1N R4,#0FCO; % (PA) MODE 4742 4742 4742 4742 4742 4742 4742 474	4796			8						
A796 3CD4 INB				7 %	INPUT					
479C 3DD4				8						
479C 3DD4									, , ,	, D D
479E 3DD4		3A44	OFCO			INB	RH4, #0FC0;	용	(PA) MC	
479E 3844 0FC0		3004		8		7 NO 22	D/ D12	0	(DD) MC	
47A2							R4, R13;			
### ### ### ### ### ### ### ### ### ##				9 9		8908	NA, HOLCO,			
47A2										
47A2 3AD8 0928				-	INPUT	AND DEC	CREMENT			
47AA 4 47AA 8 47AA 9 47AB 9 47AB 8 47AB 9 47										
47AA	47A2	3AD8	0928			INDB	R2^,R13,R9;	%	(IR, PR)	MODE
47AA	47A6	3BD8	0928			IND	R2^,R13,R9;	8	(IR, PR)	MODE
### ### ##############################										
47AA				8		TANKS OF				
47AA 3AD8 0920 47AE 3BD8 0920 47AE 3BD8 0920 47B2 47B2 47B2 47B2 47B2 47B2 47B2 47B2				90	INPUT	, DECREM	MENT AND REPEAT	,		
## ATAE		23.00	0000	8		LUDDD	DOA DIA DO		(TD DD)	
47B2						INDRB	R2 , R13 , R9;	90	(IR, PR)	MODE
### STATE		3000	0920	96		INDR	NZ , NIJ, NJ;	0	(IR, PR)	MODE
47B2 3AD0 0928				CREMENT						
A7B2 3AD0 0928				%	INPUT	AND INC	CREMENT			
47B6				4 0 8 9						
47BA 47BA 47BA 47BA 47BA 47BA 47BA 48 8 INPUT, INCREMENT AND REPEAT 47BA 48 47BA 3ADO 0920 INIRB R2^,R13,R9; % (IR,PR) MODE 47BE 3BDO 0920 INIR R2^,R13,R9; % (IR,PR) MODE 47C2 47C2 47C2 47C2 47C2 47C2 47C2 47C2								8	(IR, PR)	MODE
47BA 47BA 8 INPUT, INCREMENT AND REPEAT 47BA 8 INPUT, INCREMENT AND REPEAT 47BA 8 47BA 3ADO 0920 INIRB R2^,R13,R9; % (IR,PR) MODE 47BE 3BDO 0920 INIR R2^,R13,R9; % (IR,PR) MODE 47C2 47C2 8 0UTPUT 47C2 8 47C2 3ED4 0UTB R13,RH4; % (PR) MODE 47C4 3A46 0FC0 0UTB #0FC0,RH4; % (PA) MODE 47C8 47C8 3FD4 0UT R13,R4; % (PR) MODE 47C8 47CE 8 6 0UTPUT #0FC0,R4; % (PA) MODE 47CE 77CE 8 77CE				ONA DO				ક	(IR, PR)	
### ### ### ### ### #### #### ########				8						
47BA 3ADO 0920				8	TNDIIT	TNCDEA	AENT AND DEDEAT	920		
47BA 3ADO 0920 47BE 3BDO 0920 47C2 47C2 47C2 47C2 47C2 47C2 47C2 47C2	5 1 - 5 - 5			0	INFOI	, INCREM	TENT AND REPEAT			
47BE 3BDO 0920		3AD0	0920			TNTRB	R2^.R13.R9:	9	(TR.PR)	
47C2 47C2 47C2 47C2 47C2 47C2 47C2 47C2					, 7837	INIR	R2^,R13,R9;	96	(IR, PR)	MODE
# # # # # # # # # # # # # # # # # # #	47C2			9						
47C2 47C2 3ED4 47C4 3A46 0FC0 0UTB #0FC0,RH4; % (PR) MODE 47C8 47C8 47C8 47CA 3B46 0FC0 0UT #13,R4; % (PR) MODE 47CA 47CA 3B46 0FC0 0UT #0FC0,R4; % (PA) MODE 47CE % 47CE % 47CE 47CE 47CE 47CE 47CE 47CE 47CE 382A 09D8 0UTDB R13,R2^,R9; % (IR,PR) MODE 47D6 %										
47C2 3ED4 0UTB R13,RH4; % (PR) MODE 847C4 3A46 0FC0 0UTB #0FC0,RH4; % (PA) MODE 847C8 47C8 3FD4 0UT R13,R4; % (PR) MODE 847CA 3B46 0FC0 0UT #0FC0,R4; % (PA) MODE 847CE				8	OUTPU'	Г				
47C4 3A46 0FC0 47C8 47C8 3FD4 47CA 3B46 0FC0 47CE 47CE 47CE 47CE 47CE 47CE 47CE 3B2A 09D8 47D2 3B2A 09D8 47D6 OUT #0FC0,R4; % (PA) MODE 40FC0,R4; % (PA) MODE 41CC0,R4; % (PA)		2254								
47C8	4702	3ED4	OFCO	LINA . JUL			R13, RH4;	8		
47C8 3FD4 OUT R13,R4; % (PR) MODE 47CA 3B46 OFCO OUT #OFCO,R4; % (PA) MODE 47CE 47CE % OUTPUT AND DECREMENT 47CE 47CE 3A2A 09D8 OUTD R13,R2^,R9; % (IR,PR) MODE 47D2 3B2A 09D8 OUTD R13,R2^,R9; % (IR,PR) MODE 47D6 %				- AH. Q				6	(PA) MC	
47CA 3B46 0FC0				6				Q	(DD) MC	
47CE			OFCO					9	(PA) MC	DE DE
47CE				8			,,,,,	3	, _ , , , , , ,	
47CE	47CE									
47CE 3A2A 09D8 OUTDB R13,R2^,R9; % (IR,PR) MODE 47D2 3B2A 09D8 OUTD R13,R2^,R9; % (IR,PR) MODE 47D6 %				8	OUTPU	T AND DE	ECREMENT			
47D2 3B2A 09D8 OUTD R13,R2^,R9; % (IR,PR) MODE 47D6				8						
47D6 %										
		3B2A	09D8			OUTD	R13,R2^,R9;	ક	(IR, PR)	MODE
				9		FIRCH				

```
MACRO8000: Version 2.0 09/05/80 0.2 notateV Page 20
MACZ B:CODES L P
INPUT/OUTPUT
                         % OUTPUT, DECREMENT AND REPEAT
47D6
47D6
                          ક
47D6 3A2A 09D0 OTDRBO R13,R2^,R9; % (IR,PR) MODE 47DA 3B2A 09D0 OTDR R13,R2^,R9; % (IR,PR) MODE
                          8
47DE
47DE % OUTPUT AND INCREMENT %
47DE 3A22 09D8 OUTIB R13,R2^,R9; % (IR,PR) MODE 47E2 3B22 09D8 OUTI R13,R2^,R9; % (IR,PR) MODE
47E6
47E6
47E6 TABBER CHA THE % OUTPUT, INCREMENT AND REPEAT
47E6
47E6 3A22 09D0 OTIRB R13,R2^,R9; % (IR,PR) MODE 47EA 3B22 09D0 OTIR R13,R2^,R9; % (IR,PR) MODE
47EE
47EE
        TURNARA % SPECIAL INPUTATORES
47EE
47EE
47EE 3A45 0FC0 SINB RH4, #0FC0; % (PA) MODE 47F2 3B45 0FC0 SINT R4, #0FC0; % (PA) MODE
47F6
47F6
        A 3 3 3 9 0 MA TWO % SPECIAL INPUT AND DECREMENT
47F6
47F6 3AD9 0928 SINDB R2^,R13,R9; % (IR,PR) MODE 47FA 3BD9 0928 SIND R2^,R13,R9; % (IR,PR) MODE
47FE
47FE
47FE
                        % SPECIAL INPUT, DECREMENT AND REPEAT
47FE
                                 SINDRB R2^,R13,R9; % (IR,PR) MODE SINDR R2^,R13,R9; % (IR,PR) MODE
47FE 3AD9 0920
4802 3BD9 0920
4806
4806
                          % SPECIAL INPUT AND INCREMENT
4806
                          જુ
4806 3AD1 0928
480A 3BD1 0928
                                  SINIB R2, R13, R9; % (IR, PR) MODE
SINI R2, R13, R9; % (IR, PR) MODE
480E
480E
                          જ
480E
                          % SPECIAL INPUT, INCREMENT AND REPEAT
480E
                                  SINIRB R2^,R13,R9; % (IR,PR) MODE SINIR R2^,R13,R9; % (IR,PR) MODE
      3AD1 0920
480E
4812
       3BD1 0920
4816
4816
                                  EJECT:
```

```
MACR08000:
                Version 2.0 089/05/80
                                                   MACZ BECODES L P.
MACZ B:CODES L P
INPUT/OUTPUT
4816
         TAMES & SPECIAL OUTPUT WARDON
4816
4816 3A47 0FC0 SOUTB #0FC0,RH4; % (PA) MODE 481A 3B47 0FC0 SOUT #0FC0,R4; % (PA) MODE
481E
481E
                       % SPECIAL OUTPUT AND DECREMENT
481E
481E
481E 3A2B 09D8 SOUTDB R13,R2^,R9; % (IR,PR) MODE 822 3B2B 09D8 SOUTD R13,R2^,R9; % (IR,PR) MODE
4826
4826
                % SPECIAL OUTPUT, DECREMENT AND REPEAT
4826
482E
482E
                    % SPECIAL OUTPUT AND INCREMENT
482E
482E 3A23 09D8 SOUTIB R13,R2^,R9; % (IR,PR) MODE 8032 3B23 09D8 SOUTI R13,R2^,R9; % (IR,PR) MODE
4836
4836
                 % SPECIAL OUTPUT, INCREMENT AND REPEAT
4836
4836
4836 3A23 09D0 SOTIRB R13,R2^,R9; % (IR,PR) MODE
483A 3B23 09D0 SOTIR R13,R2^,R9; % (IR,PR) MODE
483E
483E

EJECT;
                              EJECT;
```

```
Version 2.0 089/05/80 0.8 notars
MACR08000:
                                                               Page 22
                                                        MACZ BECODES L P
MACZ B: CODES L P
PROGRAM CONTROL
                         TITLE 'PROGRAM CONTROL':
483E
                         % SYSTEM CALL 8
483E
                         % LAB2:
483E
483E
                         % DEFINED FOR (RA) OPERANDS
483E
483E
                         % CALL TOSTS
483E
                         용
                         LAB2: CALL R2^; % (IR) MODE CALL LAB; % (DA) MODE CALL LAB(R1); % (X) MODE
483E 1F20
4840 5F00 4300
4844 5F10 4300
4848
4848
4848
                         % CALL RELATIVE
4848
4848 D006
                                CALR LAB2; % (RA) MODE
484A
484A
484A
                         % DECREMENT AND JUMP IF NONZERO
484A
484A FF07
484C F788
                               DBJNZ RL7,LAB2; % (RA) MODE DJNZ R7,LAB2; % (RA) MODE
484E
                         8
484E
484E
                         % INTERRUPT RETURN
484E
484E 7B00
                          IRET;
4850
4850
4850
                         % JUMP
4850
                                      NZ,R2^; % (IR) MODE
R2^; % (IR) MODE
NZ,LAB; % (DA) MODE
LAB; % (DA) MODE
NZ,LAB(R1); % (X) MODE
LAB(R1); % (X) MODE
4850 1E2E
                                 JP
4852
       1E28
                                 JP
                                 JP
4854 5E0E 4300
                                 JP
4858 5E08 4300
485C 5E1E 4300
4860 5E18 4300
                                 JP
                                 JP
4864
                         90
4864
4864
                         % JUMP RELATIVE
4864
4864 EEEC
                                         NZ,LAB2; % (RA) MODE
LAB2; % (RA) MODE
                                 JR
4866 E8EB
                                 JR
4868
                         90
4868
4868
                         % RETURN
4868
4868 9E0E
                                 RET
                                         NZ;
486A 9E08
                                  RET;
486C
486C
                                  EJECT:
```

MACRO800 MACZ B: PROGRAM	CODES	L	Version 2.0	9/05/80		S:CODE	
186C 186C 186C 186C 7	F2C		& SYSTEM & SYSTEM &	M CALL			
186E				EJECT;			
				CALL			
			LABZ;				
							84A 84C 84E

MACZ	B:CODES L P ONTROL	Version 2.0 089/05/80 0.2 notateV				
486E		TITLE 'CPU CONTROL';				
486E		8				
86E		% COMPLEMENT FLAGS				
486E		8				
486E	8DC5	COMFLG CY, ZR;				
4870		8 MULTI-MICRO REQUEST 8				
4870		•				
4870 4870		% DISABLE INTERRUPT				
4870	7C01	DI VI;				
4872	7001					
4872		* MULTI-MICRO RESETS 8				
4872		% ENABLE INTERRUPT				
4872		96				
4872	7C04	EI NVI, VI;				
4874		% MULTI-MICRO SET . 8				
4874		8				
4874		% HALT				
4874		8				
4874	7A00	HALT;				
4876		% MO OFERATION %				
4876 4876		% LOAD CONTROL REGISTER				
4876		2 LOAD CONTROL REGISTER				
4876	7DCA	LDCTL FCW,R12;	9	(R)	MOD	E
4878		8 RESET FLACS 8		(11)	1102	32
4878		96				
4878		% LOAD FROM CONTROL REGISTER				
4878		96				
4878	7DC2		8	(R)	MOD	E
487A		8 SET FLAGS				
487A 487A		8 LOAD BLAC DYME				
487A		% LOAD FLAG BYTE				
487A	8C79	LDCTLB FLAGS, RH7;	9	(D)	MOD	F
487C	0075		70	(11)	HOD	
487C		8 - QM3				
487C		% LOAD FROM FLAG BYTE				
487C		8				
487C	8C71	LDCTLB RH7, FLAGS;	8	(R)	MOD	E
487E		8				
487E		8				
487E		% LOAD PROGRAM STATUS				
487E 487E	3020	§	0	(TD)	MOD	D
487E 4880	3920 7900 4300	LDPS R2^; LDPS LAB;		(IR) (DA)		
4884	7910 4300	LDPS LAB;		(X)	MOD	
4888	, 510 4500	& EDFS LAD(KI);	0	(A)	PIOD	L
4888		EJECT; %				

MACRO80 MACZ B CPU CON	:CODES	L P	Version 2.0 9/05/80 0.8 801218V	Page 2!
4888			% MULTI-MICRO TEST	
4888			9	
4888	7B0A		MBIT;	
488A			COMPLC CY, ZR 8	
488A 488A				
488A			% MULTI-MICRO REQUEST	
	7BCD		MREQ R12;	
488C			8	
488C			% IV IC	
488C			% MULTI-MICRO RESET	
488C 488C	7B09		% MRES; ALBAMA #	
488E	, 505		0	
488E			§IV, IVW	
488E			% MULTI-MICRO SET	
488E 488E	7B08		MSET. TJAH 8	
488E 4890	7808		MSET;	
4890			& (TJAH	
4890			% NO OPERATION	
4890	0505		% TOTAL COMMON TANK	
1892	8D07		A LOAD CONT, GON REGISTER	
4892			LDCTL FCW, Rig;	1876
4892			% RESET FLAGS	
4892	05.40		9 1097402 2097 2401 8	
4892 4894	8D43		RESFLG ZR;	
4894			LDCTL R12, PC	
4894			% SET FLAGS	
4894	0071		8	
4894 4896	8D71		SETFLG ZR,SGN,OV;	
4896			LDCTLB FLAGS, 887;	
4896			END.	
			arra para noma anda a	
			LDPS R2^;	

		186,53	- GQA		
		RH6, R21;			
		RH7,5;			
		87,5;			
		87,827;			
		RH7,R27;			
		RG R2	ENDIV B		
			ENDIX B	т.	
			STRUCTION SE		
		Numeric Lis	sting by Opcod	е	
		82',5;			
		R2";			
		R12",5;			
			Tens		
		R12", R2";	HEUG		
		82",812";			
		EB6,5;	ADDL		
			JGGA		
		1 28 088			
3 dow					
			JTJUM		
			MULTL		

```
0026
                            ADDB
                                     RH6, R2^;
                                                       % (IR) MODE
0106 0005
                            ADD
                                     R6,5;
                                                       % (IM) MODE
                                                       % (IR) MODE
0126
                            ADD
                                     R6, R2^;
0206 0505
                            SUBB
                                     RH6,5;
                                                       % (IM) MODE
0226
                            SUBB
                                     RH6, R2^;
                                                       % (IR) MODE
0306 0005
                            SUB
                                     R6,5;
                                                       % (IM) MODE
0326
                                     R6, R2^;
                            SUB
                                                       % (IR) MODE
                                     RH7,5;
0407 0505
                            ORB
                                                       % (IM) MODE
0427
                            ORB
                                     RH7,R2^;
                                                       % (IR) MODE
0507 0005
                            OR
                                     R7,5;
                                                       % (IM) MODE
                                     R7, R2^;
0527
                            OR
                                                       % (IR) MODE
0607 0505
                                     RH7,5;
                                                       % (IM) MODE
                            ANDB
0627
                                     RH7, R2^;
                            ANDB
                                                       8
                                                          (IR) MODE
0707 0005
                            AND
                                     R7,5;
                                                       % (IM) MODE
0727
                                     R7, R2^;
                            AND
                                                       % (IR) MODE
                                     RH7,5;
0807 0505
                            XORB
                                                       % (IM) MODE
                                     RH7, R2^;
0827
                            XORB
                                                       % (IR) MODE
0907 0005
                            XOR
                                     R7,5;
                                                       % (IM) MODE
0927
                            XOR
                                     R7, R2^;
                                                       % (IR) MODE
0A06 0505
                            CPB
                                     RH6,5;
                                                       % (IM) MODE
0A26
                            CPB
                                     RH6, R2^;
                                                       % (IR) MODE
OB06 0005
                            CP
                                     R6,5;
                                                       % (IM) MODE
                                     R6, R2^;
0B26
                            CP
                                                       % (IR) MODE
0C20
                            COMB
                                     R2^;
                                                       % (IR) MODE
                                     R2^,5;
R2^;
                      CPB
0C21 0505
                                                       % (IR) MODE
                      NEGB
0C22
                                                       % (IR) MODE
                                     R2<sup>^</sup>;
R2<sup>^</sup>;
R2<sup>^</sup>;
0C24
                            TESTB
                                                       % (IR) MODE
0C25 0505
                            LDB
                                                       % (IR) MODE
0C26
                            TSETB
                                                       % (IR) MODE
                                     R2^;
0C28
                            CLRB
                                                       % (IR) MODE
                                     R2<sup>^</sup>;
R2<sup>^</sup>,5;
R2<sup>^</sup>;
0D20
                            COM
                                                       % (IR) MODE
0D21 0005
                            CP
                                                       % (IR) MODE
0D22
                            NEG
                                                       % (IR) MODE
                                     R2<sup>^</sup>;
R2<sup>^</sup>,5;
0D24
                            TEST
                                                       % (IR) MODE
0D25 0005
                            LD
                                                       % (IR) MODE
                                     R2^;
0D26
                            TSET
                                                       % (IR) MODE
                                     R2<sup>^</sup>;
R12<sup>^</sup>,5;
0D28
                            CLR
                                                       % (IR) MODE
ODC9 0005
                            PUSH
                                                       % (IM) MODE
1006 0000 0005
                            CPL
                                     RR6,5;
                                                       % (IM) MODE
1026
                            CPL
                                     RR6, R2^;
                                                       % (IR) MODE
11C2
                            PUSHL
                                     R12^,R2^;
                                                       % (IR) MODE
1206 0000 0005
                            SUBL
                                     RR6,5;
                                                       % (IM) MODE
                                     RR6, R2^;
1226
                            SUBL
                                                       % (IR) MODE
13C2
                            PUSH
                                     R12^, R2^;
                                                      % (IR) MODE
1406 0000 0005
                            LDL
                                     RR6,5;
                                                       % (IM) MODE
                                     RR6, R2^;
1426
                            LDL
                                                       % (IR) MODE
                                     R2^,R12^;
15C2
                            POPL
                                                       % (IR) MODE
1606 0000 0005
                            ADDL
                                     RR6,5;
                                                       % (IM) MODE
                                     RR6, R2^;
1626
                            ADDL
                                                       % (IR) MODE
17C2
                            POP
                                     R2^, R12^;
                                                       % (IR) MODE
                                     RQ8,5;
1808 0000 0005
                            MULTL
                                                       % (IM) MODE
1828
                                     RQ8, R2^;
                            MULTL
                                                       % (IR) MODE
1908 0005
                            MULT
                                     RR8,5;
                                                       % (IM) MODE
                                     RR8, R2^;
1928
                            MULT
                                                       % (IR) MODE
1A08 0000 0005
                            DIVL
                                     RQ8,5;
                                                       % (IM) MODE
```

```
1A28 (99,91) 8
                DIVL
                           RQ8, R2^;
                                        % (IR) MODE
                OIV IN
1B08 0005
                           RR8,5;
                                        % (IM) MODE
                           RR8, R2^;
                DIV
                                        % (IR) MODE
1B28 (93 93)
1C21 0805
                                        % (IR) MODE
                LDM I
                           R8, R2^, 6;
                 TESTL
1C28 (00 (A9)
                           R2^;
                                        % (IR) MODE
1C29 0805
                LDM
                           R2^, R8, 6;
                                        % (IR) MODE
1D26 (A9)
                 LDL
                           R2^, RR6;
                                        % (IR) MODE
1E28
                88 . E JP 28
                           R2^;
                                        % (IR) MODE
1E2E (89.91)
                REALE JP ST
                           NZ, R2;
                                        % (IR) MODE
1F20 (99 91)
              LAB2: CALL
                           R2^; 8
                                        % (IR) MODE
2026
                LDB
                           RH6, R2^;
                                          (IR) MODE
2106 0005
                R2 DJ 3, R9;
                           R6,5;
                                        8
                                         (IM) MODE
210B 4300
                CL S. R.S.
                           R11, LAB;
                                        % (DA) MODE
2126 (89,87) 8
                82 dl 3 89;
                           R6, R2^;
                                        % (IR) MODE
2206 0400
                RESB
                           RH4, R6;
                                        % (R) MODE
2220
                    RESB
                           R2^,0;
                                        % (IR) MODE
2306 0400
                RES
                           R4, R6;
                                        % (R) MODE
                           R2^,0;
2320 (84 84)
                RES
                                        % (IR) MODE
                           RH4, R6;
2406 0400
                SETB
                                        % (R) MODE
2420
                           R2^,0;
                SETB
                                        % (IR) MODE
2506 0400
                SET
                           R4, R6;
                                        % (R) MODE
2520
                CA SET IA
                           R2^,0;
                                        % (IR) MODE
                BITB
2606 0400
                           RH4, R6;
                                        % (R) MODE
2620 004 (49)
                 BITB
                           R2^,0;
                                        % (IR) MODE
                 BIT
2706 0400
                           R4, R6;
                                        8
                                         (R) MODE
2720 (49)
                 BIT
                           R2^,0;
                                        % (IR) MODE
                           R2^,4;
R2^,4;
2823 (99,93)
                199 INCB
                                        % (IR) MODE
                INC
                                        % (IR) MODE
2923
2A2B
                DECB
                           R2^,12;
                                        % (IR) MODE
                           R2^,12;
2B2B
                DEC
                                        % (IR) MODE
2C26
                EXB
                           RH6, R2^;
                                        % (IR) MODE
                                        % (IR) MODE
2D26 (89,81) $
                EX
                           R6, R2^;
2E26
                LDB
                           R2^, RH6;
                                        % (IR) MODE
2F26
                RE DIS RE
                           R2^, R6;
                                        % (IR) MODE
3006 FF1C
                   LDRB
                           RH6, LAB;
                                        % (RA) MODE
3026 0014
                  LDB
                           RH6, R2^(20);
                                        % (BA) MODE
3106 FF18
                   LDR
                           R6, LAB;
                                        % (RA) MODE
                           R6, R2^(20);
3126 0014
                    LD
                                        % (BA) MODE
3206 FF10
                  LDRB
                           LAB, RH6;
                                        8
                                          (RA) MODE
3226 0014
               LDB
                           R2^(20), RH6;
                                        90
                                          (BA) MODE
3306 FFOC
                                          (RA) MODE
                   LDR
                           LAB, R6;
                                        8
3326 0014
                (LR) LD . 88
                           R2^(20), R6;
                                        ક
                                          (BA) MODE
340B FF04
                           R11, ^LAB;
                                        % (RA) MODE
                  LDR
342B 0014
3506 FF14
               (IH) & LD HA
                           R11, (R2 (20)); % (BA) MODE
                                        % (RA) MODE
                   LDRL
                           RR6, LAB;
3526 0014
                (LA) SLDL 38
                           RR6, R2^(20);
                                        % (BA) MODE
3706 FF08
                           LAB, RR6;
                  ALDRL
                                        % (RA) MODE
3726 0014
               (LH) EALDL HS
                           R2^(20), RR6;
                                        % (BA) MODE
3920 (Ad) #
                  LDPS
                           R2^;
                                        % (IR) MODE
3A22 09D8
                OUTIB
                           R13, R2^, R9;
                                        % (IR, PR) MODE
3A22 09D0
                           R13,R2^,R9;
                  OTIRB
                                        % (IR, PR) MODE
3A23 09D0
               SOTIRB
                           R13, R2^, R9;
                                        % (IR, PR) MODE
3A23 09D8
                           R13, R2^, R9;
                  SOUTIB
                                        % (IR, PR) MODE
3A2A 09D8
               OUTDB
                           R13, R2^, R9;
                                        % (IR, PR) MODE
```

```
R13,R2^,R9;
                 OTDRB
3A2A 09D0
                                          % (IR, PR) MODE
3A2B 09D8
                            R13, R2^, R9;
                     SOUTDB
                                          % (IR, PR) MODE
                            R13, R2^, R9;
3A2B 09D0
                  SOTDRB
                                          % (IR, PR) MODE
3A44 OFCO
                  INB
                            RH4, #OFCO;
                                          % (PA) MODE
3A45 OFC0
                     SINB
                            RH4, #0FC0;
                                          % (PA) MODE
                  OUTB
3A46 OFCO
                            #0FC0, RH4;
                                         % (PA) MODE
                  SOUTB
3A47 OFCO
                            #0FC0,RH4;
                                          % (PA) MODE
                  INIB
3AD0 0928
                            R2^, R13, R9;
                                         % (IR, PR) MODE
                            R2<sup>^</sup>,R13,R9;
R2<sup>^</sup>,R13,R9;
R2<sup>^</sup>,R13,R9;
                   INIRB
3AD0 0920
                                          % (IR, PR) MODE
3AD1 0928
                     SINIB
                                          % (IR, PR) MODE
3AD1 0920
                     SINIRB
                                          % (IR, PR) MODE
                  INDRB
3AD8 0920
                            R2^,R13,R9;
                                          % (IR, PR) MODE
3AD8 0928
                            R2^,R13,R9;
R2^,R13,R9;
                     INDB
                                          % (IR, PR) MODE
3AD9 0920 AG
                  SINDRB
                                          % (IR, PR) MODE
                  SINDB
3AD9 0928
                            R2^,R13,R9;
                                          % (IR, PR) MODE
                  OUTI
3B22 09D8
                            R13,R2^,R9;
                                          % (IR, PR) MODE
3B22 09D0
                            R13,R2^,R9;
                     OTIR
                                          % (IR, PR) MODE
                            R13, R2^, R9;
3B23 09D0
                    SOTIR
                                          % (IR, PR) MODE
3B23 09D8
                    SOUTI
                            R13, R2^, R9;
                                          % (IR, PR) MODE
                 OUTD
3B2A 09D8
                            R13, R2^, R9;
                                          % (IR, PR) MODE
3B2A 09D0
                            R13, R2^, R9;
                    OTDR
                                          % (IR, PR) MODE
                            R13,R2<sup>^</sup>,R9;
R13,R2<sup>^</sup>,R9;
3B2B 09D0
                    SOTDR
                                          % (IR, PR) MODE
3B2B 09D8
                    SOUTD
                                          % (IR, PR) MODE
                  IN A MI
3B44 OFC0
                            R4, #0FC0;
                                         % (PA) MODE
3B45 0FC0
                     SIN
                                          % (PA) MODE
                            R4, #0FC0;
3B46 OFC0
                     OUT
                            #0FC0,R4;
                                         % (PA) MODE
                 SOUT
3B47 OFCO
                            #0FC0,R4;
                                         % (PA) MODE
3BD0 0928
                   INI
                            R2^,R13,R9;
                                         % (IR, PR) MODE
3BD0 0920
                     INIR
                            R2^,R13,R9;
                                         % (IR, PR) MODE
                 SINIR
3BD1 0920
                            R2^,R13,R9;
                                         % (IR, PR) MODE
                SINI
3BD1 0928
                            R2^,R13,R9;
                                         % (IR, PR) MODE
3BD8 0920
                 INDR
                            R2^,R13,R9;
                                         % (IR, PR) MODE
3BD8 0928
                   IND
                            R2^,R13,R9;
                                         % (IR, PR) MODE
3BD9 0920
                SINDR
                            R2^,R13,R9;
                                         % (IR, PR) MODE
3BD9 0928
                  SIND
                            R2^, R13, R9;
                                         % (IR, PR) MODE
3CD4
                     INB
                            RH4, R13;
                                          % (PR) MODE
3DD4
                  IN
                            R4, R13;
                                          % (PR) MODE
               OUTB
3ED4
                            R13, RH4;
                                         % (PR) MODE
3FD4
                   OUT
                            R13, R4;
                                          % (PR) MODE
4006 4300
               ADDB
                            RH6, LAB;
                                          % (DA) MODE
4016 4300
                ADDB
                            RH6, LAB (R1);
                                          % (X) MODE
4106 4300
                ADD
                            R6, LAB;
                                            (DA) MODE
                                          8
4116 4300
                ADD
                            R6, LAB (R1);
                                          % (X) MODE
4206 4300
                SUBB
                            RH6, LAB;
                                          % (DA) MODE
4216 4300 SUBB
                            RH6, LAB (R1);
                                          90
                                            (X) MODE
                   SUB
4306 4300
                                            (DA) MODE
                            R6, LAB;
                                          8
               SUB
4316 4300
                            R6, LAB(R1);
                                          8
                                            (X) MODE
4407 4300
                ORB
                            RH7, LAB;
                                          % (DA) MODE
4417 4300
               ORB
                            RH7, LAB (R1);
                                          % (X) MODE
4507 4300
                     OR
                            R7, LAB;
                                          % (DA) MODE
4517 4300
                OR
                            R7, LAB(R1);
                                          % (X) MODE
4607 4300
                     ANDB
                            RH7, LAB;
                                          % (DA) MODE
4617 4300
                     ANDB
                            RH7, LAB (R1);
                                         % (X) MODE
4707 4300
               AND
                            R7, LAB;
                                         % (DA) MODE
4717 4300
                AND
                            R7, LAB(R1);
                                         % (X) MODE
```

```
% (DA) MODE
                                                                                                                                       % (X) MODE
                                     XORD RH/, LAB(R1); % (X) MODE
XOR R7, LAB; % (DA) MODE
XOR R7, LAB(R1); % (X) MODE
XOR R7, LAB(R1); % (X) MODE
XORD RH6, LAB; % (DA) MODE
XORD RH6, LAB(R1); % (X) MODE
  4907 4300
  4917 4300
  4A06 4300
  4A16 4300
 4B06 4300 CP JTR6, LAB; % (DA) MODE
4B16 4300 R6, LAB (R1); 6 % (X) MODE
4C00 4300 LAB; (DA) MODE
                                                     CP R6, LAB; % (DA) MODE
 4C01 4300 0505 CPB LAB,5;
4C02 4300 NEGB LAB;
4C04 4300 TESTB LAB;
                                                                                                                                      % (DA) MODE
                                                                                                                                      00% (DA) MODE
4C02 4300
4C04 4300
4C04 4300
4C05 4300 0505
4C06 4300
4C08 4300
4C08 4300
4C08 4300
4C08 4300
4C10 4300
4C11 4300 0505
4C11 4300 0505
4C12 4300
4C12 4300
4C14 4300
4C15 4300
4C16 4300
4C18 4300
4C18 4300
4C18 4300
4C18 4300
4C19 4300
4C10 4300
4C10 4300
4C11 4300 0505
4C12 4300
4C14 4300
4C15 4300 0505
4C16 4300
4C16 4300
4C17 4300 0505
4C18 4300
4C18 4
                                                                                                                                  % (DA) MODE
 4D18 4300 CLR LAB(R1); % (X) MODE

5006 4300 CPL RR6, LAB; % (DA) MODE

5016 4300 CPL RR6, LAB(R1); % (X) MODE
                                       PUSHL R12, LAB;
PUSHL R12, LAB(R1);
  51C0 4300
                                                                                                                                            % (DA) MODE
                                      FUSHL TRI2^,LAB(R1);
SUBL RR6,LAB;
SUBL RR6,LAB(R1).
  51C1 4300
                                                                                                                                          % (X) MODE
 5206 4300
                                                                                                                                          % (DA) MODE
% (X) MODE
 5216 4300
53C0 4300
53C1 4300
5406 4300 (IA) BAJ LDL
LDL
POPI
                                       R12^,LAB;
RAJ, PUSH R12^,LAB(R1);
                                                                                                                                            % (DA) MODE
                                                                                                                                           % (X) MODE
                                            ((IH) BAJ d LDL RR6, LAB;
gad, LDL RR6, LAB(R1);
                                                                                                                                             % (DA) MODE
                                                                                                                                          % (X) MODE
                                            POPL
                                                                                                                                          % (DA) MODE
                                                                                             LAB, R12^;
                                                                                          LAB(R1),R12<sup>*</sup>;
RR6,LAB;
  55C1 4300
                                                                                                                                            % (X) MODE
                                                                                                                                           % (DA) MODE
                                           ADDL ADDL
  5606 4300
                                          ADDL
 5616 4300
                                                                                                 RR6, LAB(R1);
                                                                                                                                               % (X) MODE
                                                                                                                                           % (DA) MODE
  57C0 4300
                                             TORRELA POP
                                                                                                 LAB, R12^;
                                                                                                 LAB(R1), R12^; % (X) MODE
  57C1 4300
                                                                         POP
                                           MULTL RQ8, LAB; % (DA) MODE RQ8, LAB (R1); % (X) MODE MULTL RR8, LAB; % (DA) MODE
  5808 4300
  5818 4300
  5908 4300
```

```
5918 4300
                    MULT RR8, LAB(R1);
                                                                % (X) MODE
 5A08 4300
                    DIVL RQ8, LAB;
                                                              % (DA) MODE
                   BALDIVL
                                       RQ8, LAB(R1);
                                                               % (X) MODE
 5A18 4300
                     (IR) BALL DIV RR8, LAB;
 5B08 4300
                                                                % (DA) MODE
5B18 4300 PAL DIV
5C01 0805 4300 PAL LDM
                                       RR8, LAB(R1);
R8, LAB, 6;
                                                               % (X) MODE
                                                               % (DA) MODE
5C08 4300 TESTL LAB;
                                                                % (DA) MODE
LDM R8,LAB(R1),6; % (DA) MODE

LDM R8,LAB(R1),6; % (X) MODE

SC19 0805 4300 LDM LAB(R1); % (X) MODE

5D16 4300 LDL LAB,RR6; % (DA) MODE

5D16 4300 LDL LAB(R1),RR6; % (X) MODE

5E08 4300 JP LAB; % (DA) MODE

5E0E 4300 JP NZ,LAB; % (DA) MODE

5E18 4300 JP NZ,LAB; % (X) MODE

5E10 4300 CALI
5C09 0805 4300 (19) 8AJ LDM LAB, R8, 6;
                        CALL LAB(R1);
5F10 4300
                                                               % (X) MODE
                LDB RH6, LAB; % (X) MODE

LDB RH6, LAB (R1); % (X) MODE

LD R6, LAB; % (DA) MODE

LD R6, LAB (R1); % (X) MODE

LD R6, LAB (R1); % (X) MODE

RESB LAB, 0; % (DA) MODE

RES LAB, 0; % (DA) MODE

RES LAB (R1), 0; % (X) MODE

RES LAB (R1), 0; % (X) MODE

SETB LAB, 0; % (DA) MODE

SETB LAB (R1), 0; % (X) MODE

SETB LAB, 0; % (DA) MODE

SETT LAB, 0; % (DA) MODE
6006 4300
 6016 4300
 6106 4300
6116 4300
 6200 4300
 6210 4300
 6300 4300
6310 4300
6400 4300
6410 4300
6500 4300
                   (DA) SET LAB, 0; 200% (DA) MODE
                                                           % (X) MODE
% (DA) MODE
6510 4300
                   LAB(R1),0;
6600 4300
                  BITB LAB(R1),0;
BIT LAB,0;
BIT LAB(R1),0;
INCB LAB,4;
INCB LAB(R1),4;
 6610 4300
                                                           % (X) MODE
                                       LAB,0; % (DA) MODE
LAB(R1),0; % (X) MODE
LAB,4; % (DA) MODE
LAB(R1),4; % (X) MODE
LAB,4; % (DA) MODE
LAB,4; % (DA) MODE
LAB(R1),4; % (X) MODE
                                       LAB, 0;
 6700 4300
6710 4300
 6803 4300
 6813 4300
                  6903 4300
6913 4300
                  INC LAB(R1),4;
6A0B 4300
                  DECB LAB(R1),12;
 6A1B 4300
                                        LAB, 12; % (DA) MODE
LAB(R1), 12; % (X) MODE
RH6, LAB; % (DA) MODE
 6B0B 4300
 6B1B 4300
                   BAJ DEC
                   (IA) BAJ EXB
 6C06 4300
6C16 4300 EXB
6D06 4300 EX
                                        RH6, LAB(R1);
R6, LAB;
                                                             % (X) MODE
% (DA) MODE
                   EX R6, LAB(R1);
LDB LAB, RH6;
                                                               % (X) MODE
 6D16 4300
6E06 4300
                                                                % (DA) MODE
6E16 4300 LAB (R1), RH6;
6F06 4300 LAB, R6;
                                                               % (X) MODE
                                                                % (DA) MODE
                   TAB(R1), 812°
                                       LAB(R1), R6;
 6F16 4300
                                                                % (X) MODE
 7026 0100
                   LDB
                   RH6,R2^(R1); % (BX) MODE
                                       RH6, R2^(R1);
                                                                % (BX) MODE
7126 0100
 7226 0100
                   LDB
                                       R2^(R1),RH6;
                                                               % (BX) MODE
```

```
7326 0100
                                  LD R2^(R1), R6; % (BX) MODE
                                   LD R11, (R2 (R1)); % (BX) MODE LDL RR6, R2 (R1); % (BX) MODE
742B 0100
7526 0100
                                  LD SLD
7726 0100
7900 4300
7910 4300
761B 4300
                                                         Rll, LAB(Rl): % (X) MODE
                                 LDL R2^(R1), RR6; % (BX) MODE LDPS LAB; % (DA) MODE LDPS LAB(R1); % (X) MODE HALT;
7A00
                              AHA IRET; 801
7B00
7B08
                                     MRES; 8838
MBIT; 838
7B09
                              MBIT;
MREQ R12;
DI VI;
EI NVI,VI;
LDCTL R12,FCW; % (R)
LDCTL FCW,R12; % (R)
ADDB RH6,RH4; % (R)
ADD R6,R4; % (R)
SUBB RH6,RH4; % (R)
7B0A
7BCD
7C01
7C04
7C04
7DC2
7DCA
7F2C
8046
8146
8246
8346
                                                                                                       MODE
                                                                                                        MODE
                                                                                                       MODE
                                ADD R6,R4; % (R) MODE SUBB RH6,RH4; % (R) MODE SUB R6,R4; % (R) MODE ORB RH7,RH4; % (R) MODE OR R7,R4; % (R) MODE AND R7,R4; % (R) MODE AND R7,R4; % (R) MODE XORB RH7,RH4; % (R) MODE XORB RH7,RH4; % (R) MODE XOR R7,R4; % (R) MODE XOR R7,R4; % (R) MODE XOR R7,R4; % (R) MODE MODE
                                                                                                       MODE
8447
8547
8647
8747
                              XORB RH7,RH4; % (R) MODE
XOR R7,R4; % (R) MODE
CPB RH6,RH4; % (R) MODE
CP R6,R4; % (R) MODE
COMB RH4; % (R) MODE
NEGB RH4; % (R) MODE
TESTB RH4; % (R) MODE
TSETB RH4; % (R) MODE
TSETB RH4; % (R) MODE
LDCTLB RH7,FLAGS;
LDCTLB FLAGS,RH7; % (R) MODE
NOP;
8847
8947
8A46
8B46
8C40
8C42
8C44
8C46
8C48
8C71
8C79
        COM R4;
NEG R4;
RESFLG ZR;
TEST R4:
TSET
8D07
8D40
                                                                                          % (R)
                                                                                                       MODE
8D42
                                                                                          % (R) MODE
8D43
                                     TEST R4;
TSET R4;
CLR R4;
8D44
                                                                                          % (R) MODE
                                                                                   % (R) MODE
8D46
                                 CLR R4; % (R) MODE SETFLG ZR,SGN,OV; COMFLG CY,ZR;
3D48
8D71
8DC5
         CPL RR6,RR4; % (R) MODE
PUSHL R12^,RR4; % (R) MODE
SUBL RR6,RR4; % (R) MODE
PUSH R12^,R4; % (R) MODE
LDL RR6,RR4; % (R) MODE
LDL RR6,RR4; % (R) MODE
POPL RR4,R12^; % (R) MODE
ADDL RR6,RR4; % (R) MODE
POP R4,R12^; % (R) MODE
9046
91C4
9246
93C4
9446
95C4
9646
97C4
```

```
9848 MULTL RQB,RR4; % (R) MODE
9948 DIVL RQB,RR4; % (R) MODE
9848 DIVL RQB,RR4; % (R) MODE
9848 DIVL RQB,RR4; % (R) MODE
9848 DIVL RRB,R4; % (R) MODE
9848 DEC RRT NZ;

A046 LDB RH6,RH4; % (R) MODE
A146 LD R6,R4; % (R) MODE
A240 RESS R4,0; % (R) MODE
A340 RES R4,0; % (R) MODE
A340 SETB R4,0; % (R) MODE
A540 BITB R4,0; % (R) MODE
A540 BITB R4,0; % (R) MODE
A540 BIT R4,0; % (R) MODE
A640 BITB R4,0; % (R) MODE
A643 INCB RH4,1; % (R) MODE
A646 BEX R6,R4; % (R) MODE
A646 BEX R6,R4; % (R) MODE
A646 TCCB ZR,RH4; % (R) MODE
A646 TCCB ZR,RH4; % (R) MODE
A646 TCCB ZR,RH4; % (R) MODE
B187 EXTSL RQB; % (R) MODE
B241 FFFE SRLB RH4,2; % (R) MODE
B242 FFFE SRRB RH4,1; % (R) MODE
B243 O900 SDLB RH4,RS; % (R) MODE
B244 RRB RH4,1; % (R) MODE
B248 FFFE SRRB RH4,1; % (R) MODE
B249 FFFE SRRB RH4,2; % (R) MODE
B249 FFFE SRRB RH4,2; % (R) MODE
B240 SDLB RH4,RS; % (R) MODE
B243 O900 SDLB RH4,RS; % (R) MODE
B244 RRB RH4,1; % (R) MODE
B245 FFFE SRRB RH4,2; % (R) MODE
B246 RCCB RRCB RH4,1; % (R) MODE
B247 FFFE SRRB RH4,2; % (R) MODE
B348 RCC RRC RA4,2; % (R) MODE
B349 FFFE SRRL RR4,2; % (R) MODE
B346 O002 SSLLB RR4,2; % (R) MODE
B347 O900 SDLL RR4,RS; % (R) MODE
B348 O900 SDLL RR4,RS; % (R) MODE
B349 FFFE SRRL RR4,2; % (R) MODE
B340 O002 SSLLB RR4,2; % (R) MODE
B341 O002 SSLLL RR4,2; % (R) MODE
B343 O002 SSLLL RR4,2; % (R) MODE
B344 FFFE SRRL RR4,2; % (R) MODE
B345 O002 SSLLL RR4,2; % (R) MODE
B346 O002 SSLLL RR4,2; % (R) MODE
B347 O900 SDLL RR4,RS; % (R) MODE
B348 O000 SDLL RR4,RS; % (R) MODE
B346 O002 SSLLL RR4,2; % (R) MODE
B347 O900 SDLL
```

2516						
B546		ADC	R6,R4;	8	(R)	MODE
B645		SBCB	RH5,RH4;	ક	(R)	MODE
B745		SBC	R5,R4;	ક	(R)	MODE
B8B0	0620	TRIB	R11^,R2^,R6;	90	(IR)	MODE
B8B2	0620	TRTIB	R11^.R2^.R6:	do	(IR)	MODE
B8B4		TRIRB	R11^,R2^,R6; R11^,R2^,R6;	00	(IR)	MODE
B8B6			RII , RZ , RO,		,	
100000000000000000000000000000000000000		TRTIRB	R11^,R2^,R6;	8	(IR)	MODE
B8B8		TRDB	R11^,R2^,R6;	90	(IR)	MODE
B8BA		TRTDB	R11^,R2^,R6;	ક	(IR)	MODE
B8BC		TRDRB	R11^,R2^,R6;	ક	(IR)	MODE
B8BE	062E	TRTDRB	R11^,R2^,R6;	90	(IR)	MODE
BA20	0765	CPIB	RH6, R2^, R7, MI;	of	(IR)	MODE
BA21	0988	LDIB	R8^,R2^,R9;	90	(IR)	
BA21		LDIRB	R8^,R2^,R9;	96	(IR)	
BA22		CPSIB	R11^,R2^,R7,NE;	90	(IR)	
BA24		CPIRB	RH6, R2^, R7, MI;	120		
BA26				8	(IR)	
		CPSIRB	R11^,R2^,R7,NE;	8	(IR)	
BA28		CPDB	RH6,R2^,R7,MI;	90	(IR)	MODE
BA29		LDDB	R8^,R2^,R9;	8	(IR)	MODE
BA29	0980	LDDRB	R8^,R2^,R9;	of	(IR)	MODE
BA2A	07BE	CPSDB	R11^,R2^,R7,NE;	90	(IR)	MODE
BA2C	0765	CPDRB	RH6, R2^, R7, MI;	90	(IR)	MODE
BA2E	07BE	CPSDRB	R11^,R2^,R7,NE;	90	(IR)	
BB20		CPI	R6,R2^,R7,MI;	of	(IR)	MODE
BB21		LDIR	R8^,R2^,R9;	00	(IR)	
BB21		LDI	R8^,R2^,R9;	000		
BB22			NO , KZ , K9;		(IR)	
		CPSI	R11^,R2^,R7,NE;	8	(IR)	
BB24		CPIR	R6,R2^,R7,MI;	90	(IR)	
BB26		CPSIR	R11^,R2^,R7,NE;	8	(IR)	MODE
BB28		CPD	R6,R2^,R7,MI;	8	(IR)	MODE
BB29		LDDR	R8^,R2^,R9;	8	(IR)	MODE
BB29	0988	LDD	R8^,R2^,R9;	8	(IR)	MODE
BB2A	07BE	CPSD	R11^,R2^,R7,NE;	90	(IR)	MODE
BB2C	0765	CPDR	R6,R2^,R7,MI;	96	(IR)	MODE
BB2E	07BE	CPSDR	R11^,R2^,R7,NE;	90	(IR)	MODE
BC47		RRDB	RH7,RH4;	olo	(R)	MODE
BD48		LDK	R4,8;	96	(R)	
BE47		RLDB		90 06	,	MODE
DE4/		KLDB	RH7, RH4;	ð	(R)	MODE
C605		LDB	RH6,5;	010	(IM)	MODE
D006		CALR	LAB2;	90	(RA)	MODE
E8EB		JR	LAB2;	olo	(RA)	MODE
EEEC		JR	NZ, LAB2;	00	(RA)	MODE
					, /	
F788		DJNZ	R7,LAB2;	010	(RA)	MODE
FF07		DBJNZ	RL7, LAB2;	0/0	(RA)	
				-	1	

		R11", R2", R7, NE;		
		R6, R2", R7, MI;		



		R6, LAB; R6, LAB(RI);				
		REG, RH4;				
			agg.			
			ANDS			
		API API	PENDIX C			
		AmZ8000 IN				
		Alphabetic Li	isting by Mr	nemonic		
		R2~,0;				
2 COM						
BOM						
						4D08

```
4106 4300
                            ADD
                                     R6, LAB;
                                                       % (DA) MODE
4116 4300
                            ADD
                                     R6, LAB (R1);
                                                       8
                                                         (X)
                                                              MODE
0106 0005
                                                       % (IM) MODE
                            ADD
                                     R6,5;
0126
                            ADD
                                     R6.R2^:
                                                       8
                                                         (IR) MODE
8046
                            ADDB
                                     RH6, RH4;
                                                         (R)
                                                       8
                                                               MODE
4006 4300
                            ADDB
                                     RH6, LAB;
                                                         (DA) MODE
                                                       8
4016 4300
                            ADDB
                                     RH6, LAB(R1);
                                                       8 (X)
                                                              MODE
0026
                            ADDB
                                     RH6, R2^;
                                                       % (IR) MODE
9646
                            ADDL
                                     RR6, RR4;
                                                       % (R)
                                                               MODE
1606 0000 0005
                            ADDL
                                     RR6,5;
                                                       8
                                                         (IM) MODE
                                     RR6, R2^;
1626
                            ADDL
                                                       8
                                                         (IR) MODE
5606 4300
                            ADDL
                                     RR6, LAB;
                                                       8
                                                         (DA) MODE
5616 4300
                            ADDL
                                     RR6, LAB(R1);
                                                       90
                                                         (X)
                                                               MODE
8747
                            AND
                                     R7, R4;
                                                       8
                                                         (R)
                                                               MODE
4707 4300
                            AND
                                     R7, LAB;
                                                       90
                                                         (DA) MODE
4717 4300
                            AND
                                     R7, LAB(R1);
                                                       % (X)
                                                               MODE
0707 0005
                            AND
                                     R7,5;
                                                       % (IM) MODE
0727
                                                         (IR) MODE
                            AND
                                     R7, R2^;
                                                       8
8647
                                     RH7, RH4;
                            ANDB
                                                       80
                                                         (R)
                                                               MODE
4607 4300
                            ANDB
                                     RH7, LAB;
                                                       8
                                                         (DA) MODE
4617 4300
                            ANDB
                                     RH7, LAB (R1);
                                                       % (X)
                                                              MODE
0607 0505
                            ANDB
                                     RH7,5;
                                                       90
                                                         (IM) MODE
                            ANDB
                                     RH7, R2^;
0627
                                                         (IR) MODE
                                                       8
2706 0400
                            BIT
                                     R4, R6;
                                                       90
                                                         (R)
                                                               MODE
2720
                            BIT
                                                        (IR) MODE
                                     R2^,0;
                                                       8
A740
                            BIT
                                     R4,0;
                                                       8
                                                         (R)
                                                               MODE
6700 4300
                            BIT
                                     LAB, 0;
                                                       8
                                                         (DA) MODE
6710 4300
                            BIT
                                     LAB(R1),0;
                                                       90
                                                         (X)
                                                               MODE
2606 0400
                            BITB
                                     RH4, R6;
                                                       90
                                                         (R)
                                                               MODE
2620
                            BITB
                                     R2^,0;
                                                       8
                                                         (IR) MODE
A640
                            BITB
                                     RH4,0;
                                                         (R)
                                                       8
                                                               MODE
6600 4300
                            BITB
                                     LAB, 0;
                                                       8
                                                         (DA) MODE
6610 4300
                            BITB
                                     LAB(R1),0;
                                                       8
                                                         (X)
                                                               MODE
1F20
                   LAB2:
                            CALL
                                     R2^;
                                                       00
                                                         (IR) MODE
5F00 4300
                            CALL
                                     LAB:
                                                       90
                                                         (DA) MODE
5F10 4300
                            CALL
                                     LAB(R1);
                                                       90
                                                         (X)
                                                               MODE
D006
                            CALR
                                     LAB2:
                                                       00
                                                         (RA) MODE
0D28
                                     R2^;
                            CLR
                                                       90
                                                         (IR) MODE
8D48
                            CLR
                                     R4;
                                                       8
                                                         (R)
                                                               MODE
4D08 4300
                            CLR
                                     LAB;
                                                       90
                                                         (DA) MODE
4D18 4300
                            CLR
                                     LAB(R1);
                                                       8
                                                         (X)
                                                               MODE
8C48
                   LAB:
                            CLRB
                                                         (R)
                                                       00
                                     RH4;
                                                               MODE
4C08 4300
                            CLRB
                                     LAB;
                                                       00
                                                         (DA) MODE
4C18 4300
                                     LAB(R1);
                            CLRB
                                                       % (X)
                                                              MODE
                                     R2^;
0C28
                            CLRB
                                                       % (IR) MODE
8D40
                            COM
                                     R4;
                                                       80
                                                         (R)
                                                               MODE
4D00 4300
                            COM
                                     LAB;
                                                       8
                                                         (DA) MODE
4D10 4300
                            COM
                                     LAB(R1);
                                                       90
                                                         (X)
                                                               MODE
0D20
                            COM
                                     R2^;
                                                       8
                                                         (IR) MODE
8C40
                            COMB
                                     RH4;
                                                       % (R)
                                                              MODE
4C00 4300
                            COMB
                                     LAB;
                                                        (DA) MODE
```

```
4C10 4300
                     COMB
                                 LAB(R1);
                                                 % (X) MODE
                     COMB
                                 R2^;
0C20 OM (8)
                                                 % (IR) MODE
                     COMFLG
8DC5 OM (AC)
                                 CY, ZR;
0D21 0005
                         CP
                                 R2^,5;
                                                 % (IR) MODE
8B46
                 ((1A) 8A CP
                                                 % (R) MODE
                                 R6, R4;
                                 R6, LAB;
4B06 4300
                         CP
                                                 % (DA) MODE
4B16 4300
                     CP CP
                                 R6, LAB (R1);
                                                 % (X) MODE
OB06 0005
                         CP
                                 R6,5;
                                                 % (IM) MODE
4D01 4300 0005
                      CP
                                 LAB, 5;
                                                 % (DA) MODE
                      CP
0B26
                                 R6, R2^;
                                                 % (IR) MODE
                  (IA) B CP
4D11 4300 0005
                                                 % (X) MODE
                                 LAB(R1),5;
8A46
                                 RH6, RH4;
                         CPB
                                                 % (R)
                                                        MODE
                       CPB
0A06 0505
                                 RH6,5;
                                                 8
                                                    (IM) MODE
                     CPB
4A06 4300
                                 RH6, LAB;
                                                 % (DA) MODE
4A16 4300
                 (IM) BA CPB
                                 RH6, LAB(R1);
                                                 % (X) MODE
                  CPB
0A26
                                 RH6, R2^;
                                                 % (IR) MODE
                     CPB
4C01 4300 0505
                                                 % (DA) MODE
                                 LAB, 5;
4C11 4300 0505
                         CPB
                                 LAB(R1),5;
                                                 % (X) MODE
0C21 0505
                                 R2<sup>^</sup>,5;
R6,R2<sup>^</sup>,R7,MI;
                         CPB
                                                 % (IR) MODE
BB28 0765
                      CPD
                                                % (IR) MODE
BA28 0765
                     CPDB
                                 RH6, R2, R7, MI; % (IR) MODE
                     CPDR
BB2C 0765
                                 R6, R2, R7, MI;
                                                 % (IR) MODE
                    CPDRB
BA2C 0765
                                 RH6, R2^, R7, MI;
                                                % (IR) MODE
BB20 0765
                    CPI
                                 R6, R2^, R7, MI;
                                                 % (IR) MODE
BA20 0765
                     CPIB
                                 RH6, R2, R7, MI;
                                                % (IR) MODE
BB24 0765
                     CPIR
                                 R6, R2^, R7, MI;
                                                 % (IR) MODE
                   CPIRB
BA24 0765
                                 RH6, R2^, R7, MI; % (IR) MODE
1006 0000 0005
                                 RR6,5;
                                                 % (IM) MODE
                         CPL
9046 M
                                                 % (R) MODE
                         CPL
                                 RR6, RR4;
1026
                         CPL
                                 RR6, R2^;
                                                % (IR) MODE
5006 4300
                         CPL
                                 RR6, LAB;
                                                 % (DA) MODE
5016 4300
                   CPL
                                               % (X) MODE
                                 RR6, LAB(R1);
                                 RR6, LAB(R1); % (X) MODE
R11^,R2^,R7,NE; % (IR) MODE
BB2A 07BE
                         CPSD
BA2A 07BE
                         CPSDB
BB2E 07BE
                         CPSDR
                  CPSDRB
BA2E 07BE
BB22 07BE
                         CPSI
BA22 07BE
                         CPSIB
BB26 07BE
                         CPSIR
BA26 07BE
                         CPSIRB
                                 R11^,R2^,R7,NE; % (IR) MODE
B040 99 81)
                         DAB
                                 RH4;
                                                 % (R) MODE
FF07
                         DBJNZ
                                 RL7, LAB2;
                                                 % (RA) MODE
6B0B 4300
                         DEC
                                 LAB, 12;
                                                 % (DA) MODE
6B1B 4300
                         DEC
                                 LAB (R1), 12;
                                                % (X) MODE
2B2B
                         DEC
                                 R2^,12;
                                                % (IR) MODE
AB4B
                         DEC
                                 R4,12;
                                                % (R) MODE
6A0B 4300
                         DECB
                                 LAB, 12;
                                                 % (DA) MODE
                     DECB
6A1B 4300
                                 LAB(R1),12;
                                               % (X)
                                                        MODE
2A2B (X) 8
                                 R2^,12;
                                                % (IR) MODE
                         DECB
AA4B
                         DECB
                                                % (R)
                                 RH4,12;
                                                        MODE
7C01 (A9)
                         DI
                                 VI;
9B48
                     DIV
                                 RR8, R4;
                                                % (R) MODE
5B08 4300
                         DIV
                                 RR8.LAB:
                                                 % (DA) MODE
5B18 4300
                                 RR8, LAB(R1);
                         DIV
                                                % (X) MODE
1B08 0005
                     DIV
                                 RR8,5;
                                                 % (IM) MODE
```

```
RR8,R2<sup>2</sup>; % (IR) MODE
RQ8,RR4; % (R) MODE
RQ8,LAB; % (DA) MODE
RQ8,5; % (IM) MODE
RQ8,LAB(R1); % (X) MODE
RQ8,R2<sup>2</sup>; % (IR) MODE
R7,LAB2; % (RA) MODE
1B28 DIV
9A48 DIVL
5A08 4300 DIVL
1A08 0000 0005 DIVL
5A18 4300 DIVL
1A28 DIVL
                                  DJNZ
 F788
 7C04 (AC) 8
                                                       EI
                                                                        NVI, VI;
R6, LAB;
                                                      EX
 6D06 4300
                                                                       R6,LAB; % (DA) MODE
R6,LAB(R1); % (X) MODE
R6,R2^; % (IR) MODE
R6,R4; % (R) MODE
RH6,LAB; % (DA) MODE
RH6,LAB(R1); % (X) MODE
RH6,R2^; % (IR) MODE
RH6,R4; % (R) MODE
RR8; % (R) MODE
R8;
R9; % (R) MODE
RQ8;
                                                                                                          % (DA) MODE
                                                       EX
 6D16 4300
2D26
AD46
                                                       EX
                                                       EX
                                 EX
EXB
EXB
EXB
 6C06 4300
 6C16 4300
 2C26
                                 EXTS
EXTSB
EXTSL
 AC46
                                                       EXB
 B18A
 B180
 B187
                                                       EXTSL
 7A00 (AI) 8
                                 HALT;
                                                                       R4, #0FC0; % (PA) MODE
R4,R13; % (PR) MODE
RH4,R13; % (PR) MODE
LAB,4; % (DA) MODE
LAB,4; % (DA) MODE
R2^,4; % (IR) MODE
LAB(R1),4; % (X) MODE
R4,4; % (R) MODE
LAB(R1),4; % (X) MODE
R2^,4; % (IR) MODE
R2^,4; % (IR) MODE
R2^,4; % (IR) MODE
R2^,1; % (IR) MODE
R2^,R13,R9; % (IR,PR) MODE
     3GOM (RI) 8
                                 IN
 3B44 0FC0
                                  NIN TRANSPORT
 3DD4
                                 INB
 3CD4 (AT)
3AD0 0928
3BD0 0920
                                                       INIB
                                INIR
INIRB
 3AD0 0920
                                                       INIRB
 7B00
                                                                       NZ,R2^; % (IR) MODE
LAB; % (DA) MODE
NZ,LAB; % (DA) MODE
LAB(R1); % (X) MODE
NZ,LAB(R1); % (X) MODE
R2^; % (IR) MODE
LAB2; % (RA) MODE
NZ,LAB2; % (RA) MODE
 1E2E MARY
                                 JP
JP
                                                                                                          % (IR) MODE
 5E08 4300
                                                      JP
 5E0E 4300
 5E18 4300
                                                      JP
 5E1E 4300
1E28
                                                      JP
                                                      JP
                                                     JRIV
 E8EB
                                             I BAJ BAR
 EEEC
                                      ENDINE (EL):
 2106 0005
                                                                    R6,5; % (IM) MODE LAB,R6; % (DA) MODE
 6F06 4300
                                   LD
```

```
6F16 4300
                  ARLDSA
                                  LAB(R1), R6;
                                                  % (X) MODE
                                  R6, R2^(R1); % (BX) MODE
7126 0100
                    LD
210B 4300
                     LD
                                  R11, LAB;
                                                 % (DA) MODE
                LD BA
                                               % (BX) MODE
7326 0100
                                  R2^(R1), R6;
742B 0100 LD LD
                                  R11, (R2 (R1)); % (BX) MODE
               LD 88
2126
                                  R6,R2^;
                                                  % (IR) MODE
                    88LDSH
761B 4300
                                  R11, ^LAB(R1);
                                                  % (X) MODE
DD25 0005 LD

342B 0014 LD

A146 LD

3126 0014 LD

4D05 4300 0005 LD

3326 0014 LD

6106 4300 LD

6116 4300 LD

4D15 4300 0005 LD

2026 LDB
                                  R2^,5;
                                                   % (IR) MODE
                                  R11, ^(R2^(20)); % (BA) MODE
                                  R6,R4;
R2^,R6;
                                                  % (R) MODE
                                                  % (IR) MODE
                                  R6, R2^(20);
                                                  % (BA) MODE
                                  LAB,5;
                                                 % (DA) MODE
                                               % (BA) MODE
% (DA) MODE
                                  R2^(20), R6;
                                  R6, LAB;
                                               % (X) MODE
% (X) MODE
% (IR) MODE
                                  R6, LAB (R1);
                                  LAB(R1),5;
                                  RH6, R2^;
2026
                        LDB
                   LDB
LDB
LDB
LDB
LDB
6E06 4300
                                                % (DA) MODE 8
                                  LAB, RH6;
                                  RH6,RH6; % (X) MODE RH6,R2^(R1); % (BX) MODE R2^(R1); % (BX) MODE R4,RH6,RH4; % (R) MODE R4^2,RH6; % (IR) MODE R4^2,RH6; % (IR) MODE
6E16 4300
7026 0100
7226 0100
A046 2E26
                                  R2^, RH6;
                                                  % (IR) MODE
                                  LAB,5;
                LDB
4C05 4300 0505
                                                  % (DA) MODE
                                                % (BA) MODE
% (X) MODE
                                  RH6,R2^(20);
                      EALDB
3026 0014
4C15 4300 0505 LDB
3226 0014 LDB
                                  LAB(R1),5;
                                  R2^(20), RH6;
                                                  % (BA) MODE
C605
                      LDB
                                  RH6,5;
                                                 % (IM) MODE
0C25 0505
                      LDB
                                  R2^,5;
                                                  % (IR) MODE
               LDB
LDB
LDB
                                  RH6, LAB;
6006 4300
                                                 % (DA) MODE
                                                % (X) MODE 82 % (R) MODE
6016 4300
                                  RH6, LAB(R1);
7DC2
                   LDCTL
                                  R12, FCW;
7DCA
                         LDCTL
                                  FCW, R12;
                                                  % (R) MODE
8C71
                      LDCTLB
                                  RH7, FLAGS;
                                               % (R) MODE
8C79
                                  FLAGS, RH7;
                         LDCTLB
                                                 % (R) MODE
                                  R8<sup>^</sup>,R2<sup>^</sup>,R9;
R8<sup>^</sup>,R2<sup>^</sup>,R9;
R8<sup>^</sup>,R2<sup>^</sup>,R9;
                                               % (IR) MODE
% (IR) MODE
% (IR) MODE
BB29 0988
                          LDD
BA29 0988
                          LDDB
BB29 0980
                         LDDR
                                  R8^,R2^,R9;
R8^,R2^,R9;
R8^,R2^,R9;
R8^,R2^,R9;
R8^,R2^,R9;
                   LDDRB
LDI
                                               % (IR) MODE
% (IR) MODE
% (IR) MODE
% (IR) MODE
BA29 0980
BB21 0988
                   LDIB
LDIR
LDIRB
BA21 0988
BB21 0980
BA21 0980
                                                % (IR) MODE
BD480 (AC)
                                  R4,8;
                                                  % (R) MODE
7526 0100
                   ELDLAB (RI)
                                                % (BX) MODE
                                  RR6, R2^(R1);
7726 0100
5416 4300
                  LDL
LDL
                                  R2^(R1), RR6;
                                                  % (BX) MODE
                                  RR6, LAB(R1);
                                                  % (X) MODE
                                  RR6, LAB;
                  LDL LDL
5406 4300
                                                  % (DA) MODE
9446
3526 0014
                                  RR6, RR4;
                                                % (R) MODE
                                  RR6, R2^(20);
                                                % (BA) MODE
                  LDL
                                                % (IM) MODE
                                  RR6,5;
RR6,R2<sup>^</sup>;
1406 0000 0005
1426
                  LDLA
                                                 % (IR) MODE
                CA LDL
3726 0014
                                                % (BA) MODE
% (DA) MODE
                                  R2^(20), RR6;
5D06 4300
                  LDL
                                 LAB, RR6;
                                                % (X) MODE
5D16 4300
                  LDL
                                  LAB(R1), RR6;
```

```
R2<sup>^</sup>,RR6;
R8,LAB,6;
LAB,R8,6;
1D26
5C01 0805 4300 LDM
                                                                     % (IR) MODE
                                                                     % (DA) MODE
5C09 0805 4300 LDM

5C11 0805 4300 LDM

5C19 0805 4300 LDM

1C21 0805 LDM
                                                                     % (DA) MODE
                                               R8, LAB(R1), 6;
                                                                     % (X) MODE
                                               LAB(R1), R8, 6; % (X) MODE
                                               R8,R2<sup>^</sup>,6;
                                                                     % (IR) MODE
                       LDM LDM
1C29 0805
                                               R2^, R8, 6;
                                                                     % (IR) MODE
                      LDPS
                                                                     % (DA) MODE
7900 4300
                                               LAB;
7910 4300 LDPS
                                               LAB(R1);
                                                                   % (X) MODE

      3920
      LDPS

      3106 FF18
      LDR

      3306 FF0C
      LDR

      340B FF04
      LDR

                                               R2^;
                                                                    % (IR) MODE
                                              R2<sup>2</sup>; % (IR) MODE
R6,LAB; % (RA) MODE
LAB,R6; % (RA) MODE
R11,^LAB; % (RA) MODE
RH6,LAB; % (RA) MODE
LAB,RH6; % (RA) MODE
LAB,RR6; % (RA) MODE
RR6,LAB; % (RA) MODE
                    LDRB
LDRB
3006 FF1C
3206 FF10
3706 FF08 LDRL 3506 FF14 LDRL
7BOA MBIT;
7BCD MREQ
7B09
                                              K12;
                   MSET;
MULT
MULT
MULT
MULT
7B09
                        MRES;
                                              RR8,5; % (IM) MODE
RR8,R4; % (R) MODE
RR8,R2^; % (IR) MODE
RR8,LAB; % (DA) MODE
RR8,LAB(R1); % (X) MODE
RQ8,R2^; % (IR) MODE
RQ8,5; % (IM) MODE
RQ8,5; % (IM) MODE
RQ8,CAB; % (R) MODE
RQ8,LAB;
RQ8,LAB; % (DA) MODE
RQ8,LAB;
RQ8,LAB(R1); % (X) MODE
7B08
1908 0005
9948
                      MULT
MULT
MULT
1928
5908 4300
5918 4300
1828
                       MULTL
                      MULTL
MULTL
MULTL
1808 0000 0005
9848
5808 4300
                    NEG R2^; % (IR) MODE
NEG LAB(R1); % (X) MODE
NEG R4; % (R) MODE
NEG LAB; % (DA) MODE
NEGB RH4; % (R) MODE
NEGB LAB; % (DA) MODE
NEGB LAB(R1); % (X) MODE
NEGB R2^; % (IR) MODE
NOP;
5818 4300
                       MULTL
0D22
4D12 4300
8D42
4D02 4300
8C42
4C02 4300
4C12 4300
0C22
                                              R7,R4; % (R) MODE
R7,LAB; % (DA) MODE
R7,LAB(R1); % (X) MODE
R7,5; % (IM) MODE
R7,R2^; % (IR) MODE
RH7,RH4; % (R) MODE
RH7,R2^; % (IM) MODE
RH7,LAB; % (DA) MODE
RH7,LAB; % (DA) MODE
RH7,LAB(R1); % (X) MODE
RH7,LAB(R1); % (X) MODE
R13,R2^,R9; % (IR,PR) MODE
R13,R2^,R9; % (IR,PR) MODE
8547
4507 4300
                        OR
                        ORAA
4517 4300
                        OR
0507 0005
                        OR
0527
                       ORB
ORB
8447
0407 0505
                        ORB
0427
                       ORB
4407 4300
4417 4300
3B2A 09D0
                        OTDR
3A2A 09D0
                       OTDRB
3B22 09D0
                        OTIR
```

3A22	09D0		OTIRB	R13,R2^,R9;	8	(TR	PR) MODE
					96		
	0.000		OUT	R13,R4;		, ,	MODE
	OFCO		OUT	#0FC0,R4;	ક		MODE
	OFCO		OUTB	#0FC0,RH4;	ક	(PA)	MODE
3ED4			OUTB	R13,RH4;	8	(PR)	MODE
3B2A	09D8		OUTD	R13,R2^,R9;	ક્ર	(IR.	PR) MODE
	09D8		OUTDB	R13,R2^,R9;	96		PR) MODE
	09D8		OUTI	R13,R2^,R9;	96		PR) MODE
						,	
	09D8		OUTIB	R13,R2^,R9;	ક	(IR,	PR) MODE
			POP	R4,R12^;	ક	(R)	MODE
17C2			POP	R2^,R12^;	8	(IR)	MODE
57C0	4300		POP	LAB, R12^;	ક	(DA)	MODE
57C1	4300		POP	LAB(R1), R12^;			MODE
			POPL	RR4,R12^;	%	, ,	MODE
	4300						
			POPL	LAB(R1),R12^;		, ,	MODE
	4300		POPL	LAB,R12 [;]	ક		
15C2			POPL	R2^,R12^;	ક		MODE
	0005		PUSH	R12^,5;	ક	(IM)	MODE
53C1	4300		PUSH	R12^, LAB(R1);	ુ લ	(X)	MODE
93C4			PUSH	R12^,R4;	96	(R)	MODE
	4300		PUSH	R12^,LAB;	96		MODE
13C2			PUSH	R12^,R2^;	9		MODE
11C2				R12^,R2^;	90		
			PUSHL		-		MODE
	4300		PUSHL	R12^,LAB(R1);		, ,	MODE
	W (FE)		PUSHL	R12 ² , RR4;	ક	. ,	MODE
51C0	4300		PUSHL	R12^,LAB;	ક	(DA)	MODE
2306	0400		RES	R4, R6;	ક	(R)	MODE
2320			RES	R2^,0;	8	. ,	
			RES	R4,0;	96	, /	MODE
	4300		RES	LAB,0;	9	, ,	
	4300					, ,	
			RES	LAB(R1),0;	ક	, ,	MODE
	0400		RESB	RH4, R6;	ક	' '	MODE
A240			RESB	RH4,0;	ક	1 /	MODE
	4300		RESB	LAB,0;	ક	(DA)	MODE
6210	4300		RESB	LAB(R1),0;	%	(X)	MODE
2220			RESB	R2^,0;	ક	(IR)	MODE
8D43			RESFLG	ZR;			
9E0E			RET	NZ;			
			RET;	19008			
B340			RL RL		0	(D)	
B240				R4,1;	8		MODE
B348			RLB	RH4,1;	8	, ,	MODE
			RLC	R4;/48	ક	(/	MODE
B248			RLCB	RH4,1;	ક		MODE
BE47			RLDB	RH7, RH4;	8	(R)	MODE
B344			RR	R4,1;	9	(R)	MODE
B244			RRB	RH4,1;	8		MODE
B34C			RRC	R4,1;	%	, ,	MODE
B24C			RRCB	RH4,1;	8	,	MODE
BC47			RRDB	RH7,RH4;	90		
2017				1111//11114/	6	(11)	MODE
DZAE			CDC	DE DA		15	0306 0000
B745			SBC	R5,R4;	90	(R)	MODE
B645			SBCB	RH5,RH4;	ક	(R)	MODE
7F2C	ROOM.		SC	44;			
B34B	0900		SDA	R4, R9;	용	(R)	MODE

0226		SUBB	RH6, R2^;	2	(IR)	MODE
	4300	SUBB	RH6, LAB(R1);			
					(X)	MODE
5216	4300	SUBL	RR6, LAB(R1);	8	(X)	MODE
1226		SUBL	RR6,R2^;	8	(IR)	MODE
9246		SUBL	RR6, RR4;		(R)	MODE
	0000 0005				, ,	
		SUBL	RR6,5;		(IM)	
5206	4300	SUBL	RR6, LAB;	de	(DA)	MODE
AF46		TCC	ZR,R4;	96	(R)	MODE
AE46		TCCB	ZR,RH4;			
	1300				(R)	MODE
	4300	TEST	LAB;			MODE
0D24		TEST	R2^;	8	(IR)	MODE
4D14	4300	TEST	LAB(R1);	8	(X)	MODE
8D44		TEST	R4;	90	(R)	MODE
4C14	4300	TESTB	LAB(R1);		(X)	MODE
0C24		TESTB	R2^;			MODE
	4300					
	4300	TESTB	LAB;			MODE
8C44		TESTB	RH4;		(R)	MODE
	4300	TESTL	LAB;	8	(DA)	MODE
9C48		TESTL	RR4;	8	(R)	MODE
1C28		TESTL	R2^;	9	(TR)	MODE
5C18	4300	TESTL	LAB(R1);		(X)	MODE
	0620	TRDB	R11^,R2^,R6;		,	
	0620					MODE
		TRDRB	R11^,R2^,R6;		(IR)	
	0620	TRIB	R11^,R2^,R6;		(IR)	MODE
	0620	TRIRB	R11^,R2^,R6;	8	(IR)	MODE
B8BA	0620	TRTDB	R11^,R2^,R6;	8	(IR)	MODE
B8BE	062E	TRTDRB	R11^.R2^.R6:			MODE
B8B2	0620	TRTIB	R11^,R2^,R6; R11^,R2^,R6;			MODE
	062E	TRTIRB	R11^,R2^,R6;		(IR)	
0D26	0025		R2^;			
	4200	TSET				MODE
	4300	TSET	LAB(R1);		(X)	MODE
4D06	4300	TSET	LAB;	8	(DA)	MODE
8D46		TSET	R4;	%	(R)	MODE
8C46		TSETB	RH4;	8	(R)	MODE
4C06	4300	TSETB	LAB;			MODE
0C26		TSETB	R2^;			
	4300			6	(TK)	MODE
4010	4300	TSETB	LAB(R1);	8	(X)	MODE
0907	0005	XOR	R7,5;	0/0	(IM)	MODE
0927		XOR	R7,R2^;			MODE
4907	4300	XOR	R7, LAB;		(DA)	MODE
4917		XOR	R7, LAB(R1);		(X)	MODE
8947		XOR	R7, R4;		(R)	
4817	4300					MODE
	4300	XORB	RH7, LAB(R1);		(X)	
0827	4200	XORB	RH7,R2^;		(IR)	
	4300	XORB	RH7, LAB;	ક	(DA)	MODE
8847		XORB	RH7,RH4;	ક	(R)	MODE
0807	0505	XORB	RH7,5;	ક	(IM)	MODE
					,	

		R11 , R2 , R6; H11 , R2 , R6; B11 , R2 , R6;		
	8			
		RII, R2, R6; RII, R2, R6; RII, R2, R6; RII, R2, R6; RII, R2, R6; RII, R2, R6; RII, R2, R6;		
	8			

APPENDIX D AmZ8000 Instruction Set: Topical Index

Instruction Description	Mnemonic	Data Types	Addressing Modes	Flags Affected
Arithmetic Island Parluga Fi			Eano	nput/Output festivet
Add with Carry Add Add All	ADC	B, W	R	C, Z, S, V, D1, H1
Add War Garry	ADD	B, W, L		C, Z, S, V, D1, H1
Compare (Immediate)	CP	B, W		C, Z, S, V
Compare (Register)	CP	B, W, L	R, IM, IR, DA, X	C, Z, S, V
001117	DAB	B, W, L	A CONTRACTOR OF THE PROPERTY O	C, Z, S
Boomia rajuot Bit	DEC	B. W		
Boordingin			R, IR, DA, X	L, O, V
		W, L		C, Z, S, V
Extoria digit	EXTS	B, W, L		0, 2, 0, 1
Horomone	INC	B, W	R, IR, DA, X	Z, S, V
Multiply	MULT	W, L		C, Z, S, V
Negate	NEG	B, W	R, IR, DA, X	C, Z, S, V
Subtract with Carry	SBC	B, W	R	C, Z, S, V, D1, H1
Subtract Sub	SUB	B, W, L	R, IM, IR, DA, X	C, Z, S, V, D1, H1
Bit Manipulation A A A A BELLA A	W.B	HO		0.
Bit Test	BIT	B, W	R	Z leaT
Bit Reset (Static)	RES	B, W	R, IR, DA, X	Test Condition Code
Bit Reset (Dynamic)	RES	B, W	R	Exclusive Or
Bit Set (Static)	SET	B, W	R, IR, DA, X	regram Centrel Irisb
Bit Set (Dynamic)	SET	B, W	R	NEIR INTRICO RIBIRES
Bit Test and Set	TSET	B, W	R, IR, DA, X	Call Procedure 8
Block Transfer and String Manipulation	W G	NACO.	lot Zeeo	Call Procedure 11 sat Decrement, Jump If the
Compare and Decrement	CPD	B, W	IR	C, Z, S, V
Compare, Decrement, and Repeat	CPDR	B, W	IR	C, Z, S, V
Compare and Increment	CPI	B, W	IR	C, Z, S, V
Compare, Increment, and Repeat	CPIR	B, W	IR ®	C, Z, S, V
Compare String and Decrement	CPSD	B, W	IR	C, Z, S, V
Compare String, Decrement, and Repeat	CPSDR	B, W	ID	CZCV
Compare String and Increment	CPSI	B, W	IR another	C, Z, S, V
Compare String and Increment, and Repeat	CPSIR	B. W	IR	C, Z, S, V stato9
Load and Decrement	LDD	B. W		Potata Left Through
Load, Decrement, and Repeat	LDDR	B. W	IR	Rotate Left Digit V
Loud, Decrement, and Hopour		B. W	IR IR	Rotate Right V
Load and moromone				Rotale Right Through
Loud, moromore, and respons	LDIR	D,		Z. V I InpiR etaloR
Translate and Decrement	TRDB	В	IR	Z, V
Translate, Decrement, and Repeat	TRDRB	В	IR SIE	Shift Dynamic V, Z
Translate and morement	TRIB	В		Shift Dynamic V, X
Translate, Increment, and Repeat	TRIRB	В	IR	Shift Left Arthry, S
Translate, Test, and Decrement	TRTDB	В	IR	Z, Voigo I that thing
Translate, Test, Decrement, Repeat	TRTDRB	В	IR	Z, V MA MORI MAS
Translate, Test, and Increment	TRTIB	В	IR	Z, Vgod Hold fine
Translate, Test, Increment, and Repeat	TRTIRB	В	IR _{gre ripco} teloege	Z, Violanteni Oli ribel
CPU Control Instructions	nut not 8.4 metrosi	onic Reter to s	is S + regular memo	he special I/O ranemonic
Complement Flag	COMFLG			C2, Z2, S2, P2, V2
Disable Interrupt	DI	-		
Enable Interrupt	EI	= 1		
Halt	HALT			
Load Control Register (from register)	LDCTL	-	R	C2, Z2, S2, P2, D2, F
Load Control Register (to register)	LDCTL			
Load Program Status	LDPS		IR, DA, X	C, Z, S, P, D, H
Multi-Bit Test	MBIT			S
Multi-Micro Request	MREQ			Z, S
Multi-Micro Reset	MRES			
	MSET			
Multi-Micro Set				
Multi-Micro Set				
Multi-Micro Set No Operation Reset Flag	NOP RESFLG			- C2, Z2, S2, P2, V2

- Flag affected only for byte operation.
 Flag modified only if specified by the instruction.

AmZ8000-Instruction Set: Topical Index (Cont.)

Instruction Description	Mnemonic	Data Types	Addressing Flags Modes Affected
Input/Output Instructions3			Regular Special
Input g v a S S	(S)IN ³	B, W	IR, DA (DA) - was a may bb A
Input and Decrement	(S)IND3	B, W	IR (IR) V
Input, Decrement and Repeat	(S)INDR3	B. W	IR (IR) V
Input and Increment	(S)INI3	B, W	IR (IR) V
Input, Increment, and Repeat	(S)INIR ³	B. W	IR (IR) V
Output V 2 X AQ 81.8	(S)OUT ³	B, W	IR, DA (DA) -
Output and Decrement	(S)OUTD3	B, W	IR (IR) V
Output, Decrement, and Repeat	(S)OUTDR3	B, W	IR (IR) V note breated
Output and Increment	(S)OUTI3	B, W	IR (IR) V
Output, Increment, and Repeat	(S)OUTIR3	B, W	IR (IR) V
Logical Instructions	(0)001	8314	Negate
And in v a sign of Ad Al Mi, A	AND	B, W	R, IM, IR, DA, X Z, S, P
Complement	COM	B, W	DID DAY 7 C D
Or	OR	B, W	R, IM, IR, DA, X Z, S, P
Test	TEST	B, W, L	R, IR, DA, X Z, S, P
Test Condition Code	TCC	B, W	R – (arsi2) IsasiA ii8-
Exclusive Or	XOR	B. W	R, IM, IR, DA, X Z, S, P
Program Control Instructions	W.B	138	11, 111, 111, 111, 111, 111, 111, 111,
	CALL	138	IR, DA, X — as been fig.
Call Procedure Call Procedure Relative	CALR	TSET	IR, DA, X — ISS BAS MET HE
Decrement, Jump if Not Zero	DJNZ	B, W	RA RESIDENT RESIDENCE TO THE PERSON RESIDENCE TO THE P
Interrupt Return	IRET	- 090	- C, Z, S, P, D, H
	JP W B	-8090	
Jump Relative	JR	- 140	IR, DA, X The Street Hand Street
	RET		
Return from Procedure System Call	SC	- FIRO -0893	Compare, increment, and Repeat = Compare String and Document =
Rotate and Shift Instructions	W S	CPSDR	Compare String Decrement and Repeat Compare String and Increment
Rotate Left	RL	B, W	Compare String, Increment, and RepRit
Rotate Left Through Carry	RLC	B, W	R C, Z, S, V
Rotate Left Digit	RLDB	Baggy	R Z, S marged had
Rotate Right	RR / 8	B, W	R C, Z, S, V
Rotate Right Through Carry	RRC	B, W	R C, Z, S, V
Rotate Right Digit	RRDB	Banan	R Z, S
Shift Dynamic Arithmetic	SDA	B, W, L	R C, Z, S, V
Shift Dynamic Logical	SDL	B, W, L	R C, Z, S, V
Shift Left Arithmetic	SLA	B, W, L	R C, Z, S, V
Shift Left Logical	SLL	B, W, L	R C, Z, S, V
Shift Right Arithmetic	SRA	B, W, L	R C, Z, S, V
Shift Right Logical	SRL	B, W, L	R C, Z, S, V

Each I/O instruction has a special counterpart used to alert other devices that a Special I/O transaction is occurring.
 The special I/O mnemonic is S + regular mnemonic. Refer to section 4.6 for further details.

APPENDIX E AmZ8000 Instruction Set Opcode Мар ун иотоинтем надам, а выват

Upper Nibble (Hex),							(Hex), Upp					T - 1				
pper Instruction Byte	0	1	2	3	4	5	6	7	8	9	A	В	С	D	E	F
	ADDB	ADD	SUBB	SUB	ORB	OR	ANDB	AND	XORB	XOR	СРВ	CP	See	See	Extend	Exter
0	R←IR	R←IR	R←IR	R←IR	R←IR	R←IR	R←IR	R←IR	R←IR	R←IR	R←IR	R←IR	Table	Table	Inst	Inst
	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	1	1		
									P	- 8						
	CPL	PUSHL	SUBL	PUSH	LDL	POPL	ADDL	POP	MULTL	MULT	DIVL	DIV	See	LDL	JP	CAL
1	R←IR	IR←IR	R←IR	IR←IR	R←IR	IR←IR	R←IR	IR←IR	R←IR	R←IR	R←IR	R←IR	Table	IR←R	PC←IR	PC←
	R←IM		R←IM		R←IM	1	R←IM	171119	R←IM	R←IM	R←IM	R←IM	2			
	LDB	LD	RESB	RES	SETB	SET	вітв	BIT	INCB	INC	DECB	DEC	EXB	EX	LDB	LD
2	R←IR	R←IR	IR←IM	IR←IM	IR←IM	IR←IM	IR←IM	IR←IM	IR←IM	IR←IM	IR←IM	IR←IM	R←IR	R←IR	IR←R	IR←
	R←IM	R←IM	R←R	R←R	R←R	R←R	R←R	R←R		(slid 2)		8				
				-					9	R						-
	LDB	LD	LDB	LD	LDA	LDL	RSVD	LDL	RSVD	LDPS	See	See	INB	IN	OUTB	OU
3	R←BA	R←BA	BA←R	BA←R	R←BA	R←BA		BA←R		IR	Table	Table	R←IR	R←IR	IR←R	IR←
	LDRB	LDR	LDRB	LDR	LDAR	LDRL		LDRL		-	3	3				
	R←RA	R←RA	RA←R	RA←R	R←RA	R←RA		RA←R		SHR						
	ADDB	ADD	SUBB	SUB	ORB	OR	ANDB	AND	XORB	XOR	СРВ	СР	See	See	Extend	Exte
4	R←X	R←X	R←X	R←X	R←X	R←X	R←X	R←X	R←X	R←X	R←X	R←X	Table	Table	Inst	Ins
	R←DA	R←DA	R←DA	R←DA	R←DA	R←DA	R←DA	R←DA	R←DA	R←DA	R←DA	R←DA	1	1		
			8/ -/	8					A	CIVISIN		4				
	CPL	PUSHL	SUBL	PUSH	LDL	POPL	ADDL	POP	MULTL	MULT	DIVL	DIV	See	LDL	JP	CA
5	R←X	IR←X	R←X	IR←X	R←X	IR←X	R←X	IR←X	R←X	R←X	R←X	R←X	Table	X←R	PC←X	PC+
	R←DA	IR←DA	R←DA	IR←DA	R←DA	IR←DA	R←DA	IR←DA	R←DA	R←DA	R←DA	10	2	DA←R	PC←DA	PC←
	LDB	LD	RESB	RES	SETB	SET	BITB	BIT	INCB	INC	DECB	DEC	EXB	EX	LDB	LI
6	R←X	R←X	X←IM	X←IM	X←IM	X←IM	X←IM	X←IM	X←IM	X←IM	X←IM	X←IM	R↔X	R↔X	X↔R	X
0	R←DA	R←DA	DA←IM	DA-IM	DA←IM	DA⊷IM	DA←IM	DA⊷IM	DA⊷IM	DA←IM	DA←IM	DA←IM	R↔DA	R↔DA	DA↔R	DA
			wed -						0.00	-						
	LDB	See	LDB	LD	LDA	LDL	LDA	LDL	RSVD	LDPS	HALT	See	EI	See	RSVD	S
7	R←BX	Table	BX←R	BX←R	R←BX	R←BX	R←X	BX←R		PS←X		Table	DI	Table		
		7		Mar 3			R←DA			PS←DA		7		7		
	ADDB	ADD	SUBB	SUB	ORB	OR	ANDB	AND	XORB	XOR	СРВ	СР	See	See	Extend	Exte
8	R←R	R←R	R←R	R←R	R←R	R←R	R←R	R←R	R←R	R←R	R←R	R←R	Table	Table	Inst.	Ins
0	n-n	n-n	n-n	n-n	n←n	n-n	n-n	n←n	n+n	n-n	H+-H	1	1	Table	inst.	ins
	CPL	PUSHL	SUBL	PUSH	LDL	POPL	ADDL	POP	MULTL	MULT	DIVL	DIV	See	RSVD	RET	RS
9	R←R	IR←R	R←R	IR←R	R←R	R←IR	R←R	R←IR	R←R	R←R	R←R	R←R	Table		PC←(SP)	
		-											2			
	LDB	LD	RESB	RES	SETB	SET	BITB	BIT	INCB	INC	DECB	DEC	=W=	mu.	7000	-
A	R←R	R←R	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	R←IM	EXB R↔R	EX R↔R	TCCB R	TC
		f as	RETRIT						ORK	RIGHT I						
	DAB	EXTS	See	See	ADCB	ADC	SBCB	SBC	See	RSVD	See	See	RRDB	LDK	RLDB	RS
В		EXTSB	Table	Table	R←R	R←R	R←R	R←R	Table		Table	Table	R	R←IM	R	
	R	EXTSL	4	4					5		6	6				
		R								GVER		0				
	LDB														N. T.	-
С	R←IM								COOK I	1000						
	04: =								P P	A I						
0	CALR													1 1		
D	PC←RA								JAIG8	-						
	JR															
E	PC←RA	-										1				
	DJNZ															
F	DBJNZ															

Notes: 1. Reserved Instructions (RSVD) should not be used. The result of their execution is not defined.

2. The execution of an extended instruction will result in an Extended Instruction Trap if the EPE bit in the FCW is a zero. If the flag is a one the Extended Instruction will be executed by the EPU function.

OPCODE MAP (Cont.)

TABLE 4. UPPER INSTRUCTION BYTE

TARLE 5

3			wer Nibbl ver Instruc		B2	B3	- 5		Lower Nibble (Hex), Lower Instruction Byte	B8	1	
Extend	See Tubis	G.Y	пыя	RI-#FI	RLB	RL	CHA RI-R		BL-B BL-B	TRIB	GCIA	
		1	0		(1 bit)	(1 bit)	MAR		M-4 0 M-4	IR.	HI-HI HI-HI	
					R	R						
		5%	via	JVIC	SLLB	SLL	404		JOJ HEUPI	ompa	RUSH	
		DOT.	RESE MUSE		R	R	AIAI		Risk 1 America	RSVD	H-HEI	
					SRLB	SRL				TRTIB		
		100	590	and i	R	R	118		2 2	IR	O.	
		-9			RLB	RL	1/11		M 4 10 0		II-R	
			2		(2 bits)	(2 bits)	R-off		R-A R-A	RSVD	PA-Ho	
					R	R	1000		3			
		100			SDLB	SDL	SL-AS	UVSR	81-8 8-88	TRIRB	E-8	
			3		R	R	JROJ		RECU 4 ROLL	IR	RGJ	
					RRB	RR	R-AR		49 0 P 40		8-19	
			4		(1 bit)	(1 bit)				RSVD		
		96	50	RID	R	R	OHA X-49		ano 5 aug	ESGS.	GQ.s	
		1007			AG-0	SLLL	AC-R		R-DA R-DA	TRTIRB	K-R	
			5		RSVD	R			6	IR	100	
			710	Jyrd	7.31.03	SRLL	909		10.1 100100	tales.	PUSH	
		SET.			RRB	RR	X-PN		X-48 X-48	RSVD	K-vAir	
			6		(2 bits)	(2 bits)	AGRI	AGHARI	AC-R 7AC-RI	D-P	O-BI	
		0	- 2000	1000	R	R	THE		STEG SEES	TRDB	GJ	
		4			RSVD	SDLL	100-01		644 × 8 644 ×	IR	KB	
		HA.	MI27	1/1//(2)	MI-AU	R	MENAU		MI-40 10-40 4	- 49	G-R	
		-			RLCB	RLC				RSVD		
		2	8		(1 bit)	(1 bit)	B-30	AGJ X all .	May 9 all	80.1	Sen	
					R	R				TRTDB	X :	
					SLAB	SLA			ΑΑ	IR		
		13	9 9		SRAB	SRA	0110		685 612	1000	COA	
					R	R	. Past		B B	RSVD	8-8	
					RLCB	RLC						
		18	A		(2 bits)	(2 bits)	905		FRRH FOR	TRDRB	KEUR	
		187	FAR		R	R	RI-R		яня Сяня	IR	E-PII	
					SDAB	SDA				RSVD		
		3	ОЗОВ		R	R	118		STEE D SEE	1438	a.	
		991	28-88	Prili-	RRCB	RRC	Mark		M-8 M-9	- 15	-8	
		00	C		(1 bit)	(1 bit)	Des		Sola E sol	TRTDRB		
					R	R	F-A		Son E son	Hall In	erner	
						SLAL				3	TEXT	
			D		RSVD	R			F	RSVD	6	
						SRAL						
			-		RRCB	RRC						
			E		(2 bits)	(2 bits)	-					
					- "	-						
			F		RSVD	SDAL						
					1.040		1					

TABLE 1. UPPER INSTRUCTION BYTE TABLE 3. UPPER INSTRUCTION BYTE

8D	8C	4D	4C	OD	ос		Lower Instruction
COM R	COMB R	COM X DA	COMB X DA	IR IR X			O
SETFLG	LCTLB R←FLGS	CP X, IM DA, IM	CPB X, IM DA, IM	CP IR, IM	CPB IR, IM	MOS TAS	1
NEG R	NEGB R	NEG X DA	NEGB X DA	NEG IR	NEGB IR	17G HSHS	2
RESFLG	RSVD	RSVD	RSVD	RSVD	RSVD	9 10 11 11 11 11	3
TEST R	TESTB R	TEST X DA	TESTB X DA	TEST		1153 11606	4
COMFLG	RSVD	LD X←IM DA←IM	LDB X←IM DA←IM	L D IR←IM	LDB IR←IM	8	5
TSET R	TSETB R	TSET X DA	TSETB X DA	TSET IR	TSETB	ano-es gyi	6
NOP	RSVD	RSVD	RSVD	RSVD	RSVD	@vis	7
CLR R	CLRB R	CLR X DA	CLRB X DA	CLR	CLRB	JESH 5-49	8
	LDCTLB FLGS←R	8		PUSH	II. MA	3734 R-16	9

	Nibble (Hex), estruction Byte	3A	3B
Si		INIB	INI
	0	IR←IR	IR←IR
		INIRB	INIR
		IR←IR	IR←IR
EPP2.3		SINIB	SINI
	1	IR←IR	IR←IR
		SINIRB	SINIR
		IR←IR	IR←IR
		оитів	OUTI
	0	IR←IR	IR←IR
	2	OTIRB	OUTIR
		IR←IR	
		IH←H	IR←IR
		SOUTIB	SOUTI
	3	IR←IR	IR←IR
		SOTIRB	SOTIR
		IR←IR	IR←IR
		INB	IN
	4	R←DA	R←DA
91.7		SINB	SIN
	5	R←DA	R←DA
OVER.			
		OUTB	OUT
	6	DA←R	DA←R
E CHANGE		SOUTB	SOUT
	7	DA←R	DA←R
BUBL		INDB	IND
	8	IR←IR	IR←IR
		INDRB	INDR
		IR←IR	IR←IR
		SINDB	SIND
	9	IR←IR	IR←IR
		SINDRB	SINDR
		IR←IR	IR←IR
	a	OUTDB	OUTD
	Α	IR←IR	IR←IR
		OTDRB	OTDR
		IR←IR	IR←IR
	В	SOUTDB IB←IB	SOUTD
	В	SOTDRB	
		SOIDER	SOTDR

TABLE 2. UPPER INSTRUCTION BYTE

Lower Nibble (Hex) Lower Instruction Byte	1C	5C	9C
0	RSVD	RSVD	RSVD
A Hones ova	LDM R←IR	LDM R←X R←DA	
8	TESTL IR	TESTL X DA	TESTL R
9	LDM IR←R	LDM X←R DA←R	

OPCODE MAP (Cont.)

TABLE 6.

TABLE 7.

	Nibble (H		BA	ВВ
egn I-Bi	0	8	CPIB IR	CPI IR
946 1-01 1948 1-191	1		LDIB IR←IR LDIRB IR←IR	LDI IR←IR LDIR IR←IR
RI	2		CPSIB IR	CPSI IR
E-EI BITO I-BI	3	-	RSVD	RSVD
tuos IRi	4		CPRIB IR	CPIR
FIL	-5		RSVD	RSVD
in-fi sue	6		CPSIRB	CPSIR
TUD	7		RSVD	RSVD
900	8	4	CPDB IR	CPD IR
CS0 70 W15H 71	9	e	LDDB IR←IR LDDRB IR←IR	LDD IR←IR LDDR IR←IR
SING	А		CPSDB IR	CPSD IR
-RI.	В		RSVD	RSVD
010	С		CPDRB IR	CPDR
ude -Fe	D	9	RSVD	RSVD
-RI	E		CPSDRB	CPSDR
	F		RSVD	RSVD

	rer Nibble r Instruction		7B	7D
iĝo A	0	SM0.3	IRET PC←(SSP)	RSVD
		AG		
Ta.I	91	683	RSVD	RSVD
	2		RSVD	LDCTL R←FCW
	3	8088 X	RSVD	LDCTL R←RFRSH
ricke.	0/4	OVER	RSVD	LDCTL R← PSAPSEG
anr R	5	ETESTE X AG	RSVD	LDCTL R← PSAPOFF
	6		RSVD	LDCTL R←NSPSEG
	7		RSVD	LDCTL R←NSPOFF
	8	AG	MSET	RSVD
OR .	9	CVI	MRES	RSVD
10	A	X	мвіт	LDCTL FCW←R
	В		RSVD	LDCTL RFRSH←R
	С			LDCTL PSAPSEG ←R
	D	OURT	MREQ R	LDCTL PSAPOFF ←R
	Е		RSVD	LDCTL NSPSEG←R
	F	2 10	RSVD	LDCTL NSPOFF←R

A

ers addamuprs busmmod IIA APPENDIX Felpric s ers abusmmod IIA

The following code is taken from the Executive Module of MONITOR3, a sample program developed at the AMD Customer Education Center by Charles R. McCallan and Bruce W. Pettner. MONITOR3 was developed to demonstrate the entire AmZ8002 instruction set, MACRO8000 (MACZ) and the linker.

MONITOR3 is the principal vehicle of instruction used in the center's course ED8000B, Assembly Language Programming for the AmZ8000. A complete listing of MONITOR3 is available to students enrolling in the ED8000S or ED8000B seminars. It is reprinted in the ED8000A/B STUDY GUIDE.

MONITOR3

THIS IS A SANELE PROGRAM NOTINOM SIGNAL A CONSIBILITY IS ASSUMED BY

The following listings are for a simple monitor program for the AmZ8002. The program is written in 15 modules and linked together. The listings include the linking operation, which was done from a directives file.

This 'simple-minded' monitor was written to demonstrate the MACZ (MACRO8000) macro assembler and LNKZ (LINK8000) linker which support the AmZ8000. The code is an attempt to demonstrate various methods of programming the AmZ8000 with a consious effort to do things in as many ways as possible while maintaining good programming practices. The main intent of the program is to show example AmZ8000 code that is structured and WELL DOCUMENTED.

This monitor is NOT intended to be a sophisticated program for end users, but with suitable extentions might form the basis for a useful small monitor.

The 'simple' monitor as implemented supports the following basic commands and DEBUG functions:

ALTER OR DISPLAY MEMORY (A command) SET SOFTWARE BREAKPOINT (B command) DUMP MEMORY (D command) FCW ALTER/DISPLAY (F command) GOTO OR RESUME EXECUTION (G command) HELP (LIST COMMAND SUMMARY) (H command) PROGRAM DOWNLOAD (.BIN FILE) (L command - dummy module) MEMORY FILL (M command) DISLAY CURRENT PC (P command) REGISTER ALTER/DISPLAY (R command) DISPLAY SYSTEM STACK POINTER (S command) EXIT ANY COMMAND (ESC)

All commands are a single letter. All command arguments are either DECIMAL designated by nn or HEX designated by hhhh (i.e. G4AFO starts execution at address 4AFO). Alter/display commands always alter a full word (4 HEX digits). To exit ANY command at any time type an esape (ESC key). These conventions greatly simplify the program logic and help keep the code simple and understandable.

If it is desired to add a command the procedure is very simple. The command (single letter) needs to be added to the command table in the 'DATA' module and the entry point label must be entered in the 'ONGOTO' statement in the 'EXEC' modules command decode routine. The entries (command and label) must be in the same relative positions in both the command table and entry (label) list. The new command will then be decoded by the CCP routines and control transferred to your entry point.

Every effort has been made to comment and structure the code so that it is self-documenting. Comment blocks explain each routine and should make following the code very easy.

THIS IS A SAMPLE PROGRAM ONLY AND NO RESPONSIBILITY IS ASSUMED BY ADVANCED MICRO DEVICES OR ADVANCED MICRO COMPUTERS FOR ITS USE.

The monitor as linked will run on the AmZ8000 Evaluation Board but could also be linked for prom generation by setting the absolute assignments of the 'DATA' and 'NPSA' segments to '0' and '256' (decimal).

MACZ M3EXEC O L=B:M3EXEC.PRN W D M P

Version 2.0

9/05/80

ORS EXECUTIVE MODULE

0000 MREKEC O LEBIMBEXE & PRN WID MIP

This is a simple monitor program for the AmZ8000 EVALUATION BOARD. This program was created as an exercise in utilizing the MACRO8000 ASSEMBLER in an engineering prototype enviornment.

An attempt was made to use reasonable structured programming techniques.

- 1. Liberal comments have been used to aid program with the decimentation.
- 2. Labels and constants were chosen to make the wood program as readable as possible.
 - 3. Different structures were used throughout the program to demonstrate as many programming methods as possible.

The program is divided into routines to handle console I/O and initialization, stack initialization and routines to perform the monitor functions and commands.

The program has been developed in modules for input to the LINK8000 LINKER to demonstrate modular program development and to allow partitioning the monitor into ROM and RAM areas of memory.

Each module carries a comment block explaining its function and the interfacing protocal it requires.

ώ

0000

0000

0000

0000

0000

0000

0000

0000

0000

0000

0000

0000

0000

0000

0000

0000

0000

MACR08000: Version 2.0 9/05/80 Page 2 MACZ M3EXEC O L=B:M3EXEC.PRN W D M P MONITOR3 EXECUTIVE MODULE 0000 and the MODULE so'M3EXEC'; as in reduites 0000 0000 TITLE 'MONITOR3 EXECUTIVE MODULE'; 0000 GEASTO BUSINES SUG HEADER MONITOR3 EXECUTIVE MODULE', 0000 0000 TIAKSOOD CIMMES 40 G'4116 CONFIGURATION', OR SHE The program has been devid/50/80 Af.0; es for imput to 0000 0000 0000 PAGE 35; and routines to perform the monitor functions and 0000 Charles R. McCallan 0000 The prodram is granged into rom Bruce W. Pettner 0000 0000 Wellods as DATE: Die 0000 4/29/80 0000 0000 3" DILLEGADE VERSION & MOLE 4.0 CHROSONOGE CUE 0000 0000 Module Description tabels and constants were chosen to make the 0000 0000 This is the main module of the program. During initialization 0000 it defines the NPSA and loads the NPSAP register, initializes the System Stack Area and loads the Stack Pointer, initializes 0000 0000 the P6 Serial Port for the monitor console and calls LPUTLINE 0000 to output the initialization message and initializes the 0000 memory Refresh register. 38 1698009016 8010000169 0000 0000 The module also contains the Console Command Processor (CCP) 0000 routine which outputs a prompt and allows command input through 0000 the use of the M3PUTLN and M3GETLN subroutines, decodes the 0000 monitor commands and jumps to the appropriate command routine 0000 or error message handler. 0000 0000 The EXECUTIVE MODULE also contains the service routines for 0000 Trap and Interrupt handling. MBEXEC O L-B-MBEXE & BRN W D M P 0000

Version 2.0

	Version 2.0				Page 3		
MACZ M3EXEC O	L=B:M3EXEC.PRN W	I D M P					
MONITORS EXECT	DIIVE MODULE						
0000							
0000							
0000					LSTART,	96	Monitor entry point
0000					LWHAT,		Invalid input routin
0000					LCCP;		Normal entry for
0000							return from command
0000						96	processing
0000	do						
0000			EXTERNAL	L intro	LINITMSG,	8	Data fields
0000					LPROMPT,	8	
0000					LCMDTBL,	9	
0000					LWHATMSG,	8 mm	
0000					LOPMSG,	8	
0000					LPRMSG,	8	
0000					LEXITMSG,	9900008	
0000					LBRKMSG,		
0000	96						
0000					LALTER,	Mamor &	Entry points for
0000					LREG,	naeq e	Commands
0000					LFLAG,	50 00 M &	
0000							
0000						8 mark 8	
0000					LSSP,	CONUES 8	
0000					LHELP,	1 CM TO 8	
0000					LMFILL,	ક	
0000					LDUMP,	8	
0000					LBPSET,	8	
0000					LBRKP,	8	
0000					LOAD,	8	
0000	96						
0000					LGETLINE,		Entry Points for
0000					LPUTLINE,	8	Common subroutines
0000					LHEXVERT,	8	
0000	MATAR MODOL %	(A)					
					LSTACK.	9	Stack area of RAM

```
MACR08000: Version 2.0 9/05/80
                                                          Page 4
MACZ M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
                                                                         % Breakpoint save area
0000
                                                          LBPSAV,
0000
                                                          LSAVPS.
                                                                         % Prgm status save area
0000
                                                                         % 128 character I/O
                                                          LINEBUFF:
0000
                                                                         % buffer...word aligned
0000
0000
                             Constant Definitions:
0000
0000
                                                   = #4000, FREE % FCW for interupt or trap
                                    CONST
                                           IFCW
0000
                                                                % Counter register
                                           COUNT
                                                   = R3,
0000
                                            STACKP = R15,
                                                                 % Stack Pointer
                                                   = #OFE9,
0000
                                           CTLP6
                                                                 % Console control port
0000
                                            CTLP5
                                                   = #OFED, % P5 download port
0000
                                            BUFFB
                                                   = RL1, pygg % Used to buffer byte I/0
                                                          % Memory pointer used to
0000
                                            INDEX
                                                   = R9
0000
                                                                 % process I/O lines or
                                                          PREKMAGO % data strings
0000
                                                         Secondary memory pointer
0000
                                            SCAN
                                                   = R4.
                                                          Thinked % Used to pass data to/from
0000
                                           HEX
                                                   = R7.
                                                          roswag % HEX/ASCII conversion routines
0000
0000
                                           HEXLOW = RL7, page 28 Low byte of HEX register (R7)
                                           CNTCTL = #0FE7, 8 8253 Cntr/tmr mode port addr
0000
                                                  = #0FE5, PEROMED % 8253 Timer 1 port address
0000
                                           CNTR1
0000
                                           CNTR2 = #0FE6, 8 8253 Timer 2 port address
0000
                             Set baud rate constants as follows:
0000
0000
                                      13 = 9600 Buad
0000
                                      26 = 4800 "
0000
                                      52 = 2400 "
0000
                                     104 = 1200 "
0000
                                     417 = 300 "
0000
                                    1136 = 110 "
0000
                                                               % Baud rate constant #1
0000 TORA EXECUTIVE MODULE
                                           BAUD1 = 13,
                                           BAUD2 = 13;
0000 MAEXEC O L=BAMBEXEC.PRN W D M P
                                                                % Baud rate constant #2
0000000000
           Versign 2.0 - 9/05/80
```

LP:

END:

It is intended trb ::= NIT; de of a single letter

INC IR, 2;

LD

EJECT:

IR^, SRC;

9/05/80

Page 5

MACROSOOO:

0000

0000 0000

0000

0000

0000

0000

0000 OF EXECUTAR WODDER

Version 2.0

MACZ M3EXEC O L=B:M3EXEC.PRN W D M P

MONITOR3 EXECUTIVE MODULE

```
Page 6
MACR08000:
                 Version 2.0 9/05/80
MACZ M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
0000
0000
                                             ONGOTO MACRO
0000
                              This is a MACRO to aid in decoding a command line.
0000
                              It is intended to allow decode of a single letter
0000
                              command after the command has been converted to a
0000
0000
                              number.
0000
0000
                              The conversion may be done with a CPDRB instruction
                              and a command table by scanning for a match and using
0000
                              the relative position of the command in the table.
0000
0000
                              The ONGOTO MACRO is passed a register containing the
0000
0000
                              number or position and a list of labels in the same
                              order as the table. The result is a 'POOR MAN'S CASE
0000
0000
                              STATEMENT, . WITHOUT SCORE & MORD IN MEMORY METHOD AND LINES.
0000
                              The call for the MACRO has the form :
0000
                       0000
0000
                                      ONGOTO REG, (LAB1, LAB2, ... LABN);
0000
                                      MACRO ONGOTO X, LABLIST;
0000
0000
                                                  Aburce file for the fill macro
0000
0000
                                                   Z: OBJECT;
0000
                               INCLUDE Stateme BEGIN The macro files are:
                               cellable, they ext: = 1; eperate source liles and are called via
0000
                               SASTISPIS TO SUA FOR TO Y IN LABLIST DO OLOSE TO Wake the wactor
0000
0000
                               The following macros at BEGIN in this module as well as being
                       REBERGE BERGERS RESERVES BERGERS RESERVES IF BER X EQ Z SERRES BERGERS BERGERS BERGERS BERGERS
0000
                                                      THEN JP Y:
0000
                                                       Z ::= Z + 1
0000
                                                     END
0000 LONG EXECUTIVE MODULE
0000 M36XEC C L=B:M3EXEC PRN W D M P
                                             END;
```

T

0000

```
MACR08000: Version 2.0 9/05/80
                                                      Page 8
MACZ M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
0020
                                        SEGMENT 'CODE';
0000
                    0000
                           *** This is the Main Entry Point of the monitor program ***
0000
0000
                    LSTART: LD
                                        RO, #8000+30*512;
                                                             % 30us Refresh rate
     2100 BC00 0000
0004
                                                             % Load counter
     7D0B
                                  LDCTI.
                                        REFRESH, RO;
0006
     2100*0000
                                  LD
                                        RO, LNPSA;
                                                             % Set up the NPSAP
000A
                                  LDCTL
     7D0D
                                        PSAPOFF, RO;
000C
000C
                          This routine initializes the system stack
000C
000C
                                        INDEX, ^LSTACK;
                                                             % Set index reg to stack area
     2109*0000
                                  LD
                                        COUNT, 128;
                                                           % Load length in count reg
0010
     2103 0080
                                  LD
0014
     0D95 0000 A991
                                 FILL
                                        INDEX, 0, COUNT;
                                                            % Init Stack and Save areas
001A
     F384
                                                             % Initialize Stack Pointer
001C
     210F*0038
                                 LD
                                        STACKP, LSTACK + 56;
0020
                                 LD
                                        LSAVPS, IFCW;
                                                            % Load default FCW
     4D05*0000 4000
                                 LD
0026
     2100*0000
                                        RO, LSTART;
                                        LSAVPS(2),R0;
002A
     6F00*0002
                                 LD
                                                            % Load default PC
002E
002E
                           Set Timers for baud rates at P6 and P5 serial ports
002E
                    SMERVE
                                        RLO, #76;
                                                             % Counter 1 mode 3 (P6)
002E
     C876
                                  LDB
                                                           % Send mode command
0030
     3A86 OFE7
                                        CNTCTL, RLO:
                                  OUTB
0034
     C8B6
                           page to LDB BRLO, #B6; 9E MOTOO
                                                           % Counter 2 mode 3 (P5)
0036
     3A86 OFE7
                           OUTB CNTCTL, RLO; % Send mode command X
                           bisced aLDs wognRo, BAUD1; carron ar junk % P6 baud rate apind a
003A 2100 000D
                           CNTR1, RLO; Men Stodies 2% Send LS byte of rate
003E 3A86 0FE5
                    OUTB CNTR1,RH0; % Send MS byte of rate
0042 3A06 OFE5
0046 2100 000D
                                        RO, BAUD2; % P5 baud rate
                                  LD
                                        CNTR2, RHO;
                                                           % Send LS byte of rate
004A 3A06 OFE6
                                  OUTB
                                                           % Send MS byte of rate
004E 3A86 OFE6
                                  OUTB
                                        CNTR2, RLO;
0052 OKS EXECULTAE WODNES
                                  EJECT;
```

```
MACR08000: Version 2.0 9/05/80
                                                         Page 10
MACZ M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
0092
                            This routine outputs the initialization message courses courses
0092
0092
0092
      2109*0000
                                   LD
                                          INDEX, LINITMSG;
                                                                % Set index to message
0096
                                                                % Output message
     5F00*0000
                                   CALL
                                          LPUTLINE;
009A
                                                                % Go to Help routine to
      5E08*0000
                                   JP
                                           LHELP;
009E
                                                                % output instructions
009E
009E
009E
                            This is the Console Command Procrssor routine (CCP)
009E
009E
                            Thie routine functions as the Executive routine for the
009E
                            Montior3 program.
009E
009E
                            CCP controls prompting the user, input and decoding of monitor
009E
                            commands and calls the routines that execute the monitor commands
009E
009E
009E
                            Routine to output monitor prompt
009E
009E
      2109*0000
                     LCCP:
                                   LD
                                          INDEX, LPROMPT;
                                                                % Set index to prompt
00A2
      5F00*0000
                                   CALL
                                          LPUTLINE:
                                                                % Output prompt
                     9 ETVA
00A6
00A6
                            Routine to input monitor command
00A6
00A6
                     LMONIN: LD INDEX, LINEBUFF;
                                                                % Set index to buffer
      2109*0000
                            LDB INDEX^,79; % Set input length
OOAA
      0C95 4F4F
OOAE
      5F00*0000
                                   CALL
                                          LGETLINE;
                                                                % Input command
                            % Retry if ESC was input
00B2
      5E04*009E
00B6
00B6
                                   EJECT;
```

0100

0106 010C

EE02 5E08*0000 0A08 0707 EE02

010C 5E08*0000 0A08 0112 0808 EE02 5E08

MACR08000:

```
MACZ M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
00B6
00B6
                                     Routine to process monitor command and a second process monitor
00B6
00B6
                              Routine uses RO to decode command, RL1 (BUFFB) to hold command
00B6
                           character being decoded, R4 (SCAN) to scan command table and
00B6
                              R9 (INDEX) and R3 (COUNT) to step through command string.
00B6
                              The CPDRB instruction generates the position of the command
                              being decoded in the command table. Then the position is used
00B6
00B6
                              by the ONGOTO macro to pick a destination.
00B6
00B6
      8D08
                             essens CLR pue RO; y sug rue meet redie % Clear register O
00B8
      2104*0000
                             SCAN, LCMDTBL; % Set R4 to command table
OOBC
                                                                    % Load cmd table length
      2048
                                     LDB
                                             RLO, SCAN;
OOBE
                             AND SCAN, RO; ware the super & Set R4 to end of cmd table
      8104
00C0
      A900
                                             RO,1;
                                                                    % Adjust RO to make
00C2
                                                                    % R0=position after scanning
00C2
                       00C2
                                                                    % Step R9 past length byte
      A990
                                     INC
                                             INDEX,1;
                                             BUFFB, INDEX^;
                                                                    % Load command character
00C4
      2099
                                     LDB
00C6
      BA4C 0096
                                     CPDRB
                                             BUFFB, SCAN, RO, EQ;
                                                                   % Decode command
OOCA
OOCA
                                     ONGOTO RLO, (LALTER, LREG, LFLAG, LGO, LPC, LSSP, LBPSET, LHELP,
OOCA
      0A08 0101 EE02
                              Treds | command strot LMFILL, LDUMP, LOAD);
00D0
      5E08*0000 0A08
0006
      0202 EE02 5E08
OODC
     *0000 0A08 0303
00E2
     EE02 5E08*0000
00E8
     0A08 0404 EE02
OOEE
     5E08*0000 0A08
00F4
      0505 EE02 5E08
00FA *0000 0A08 0606
```

Page 11

Version 2.0 9/05/80

```
F-14
```

```
MACR08000: Version 2.0 9/05/80
                                                           Page 12
MACZ M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
0118 *0000 0A08 0909
011E EE02 5E08*0000
0124
      0A08 0A0A EE02
012A
      5E08*0000 0A08
0130
      OBOB EE02 5E08
0136
     *0000
0138
                             Illegal command error handler powb rowp)
0138
0138
0138
                                            INDEX, ^LWHATMSG;
                                                                   % Set R9 to '?' prompt
      2109*0000
                      LWHAT:
                                     LD
013C
      5F00*0000
                                                                   % Output prompt
                                     CALL
                                            LPUTLINE;
0140
      5E08*00A6
                                                                   % Return to monitor
                                     JP
                                            LMONIN;
0144
0144
0144
                                     Opcode Trap service routine
0144
0144
                             The following routines handle the interrupts and traps. or come repre-
0144
                             All routines output a message and save the current user
0144
0144
                             status from the stack and the user registers in the register
0144
                             save area.
0144
0144
                             The System Call routine outputs the identifier from the SC
0144
                             instruction as part of the exit message. The communication as part of the exit message.
0144
                              89 (INDEX) and R3 (COUNT) to step through command string.
0144
                      LOPCODE: LDM LSVAREA, RO, 15; % Save user regs 0-14
      5C09 000E*0000
014A
                              MODERN LDCTL NO RO, NSPOFF; Commond & Svae user R15 and and
      7D07
      6F00*001E
                                                            % Store in save area
014C
                                            LSVAREA(30),R0;
0150
      2109*0000
                                     LD norma INDEX, LOPMSG; room & Output error message
0154
      5F00*0000
                       ARREST AND ARREST CALL SE LPUTLINE;
0158
     5E08*01EC
                                     JP
                                            LRESTORE;
                                                                   % Return to monitor
015C
015C LOWS EXECULTAE WODRES.
                                     EJECT:
```

```
MACR08000:
                  Version 2.0
                                9/05/80
                                                              Page 13
MACZ M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
015C
015C
                               Privileged Instruction Trap service routine
015C
015C
       5C09 000E*0000
                       LPRIV:
                                       LDM
                                              LSVAREA, RO, 15;
                                                                      % Save user regs 0-14
0162
                                       LDCTL
                                              RO, NSPOFF;
                                                                      % Save user R15
0164
       6F00*001E
                                              LSVAREA(30), RO;
0168
0168
       2109*0000
                                       LD
                                              INDEX, ^LPRMSG;
                                                                       Output error message
016C
      5F00*0000
                                      CALL
                                              LPUTLINE;
0170
                                              LRESTORE:
                        % Return to monitor
0170
       5E08*01EC
0174
0174
                               This is the System Call handler routine....
0174
0174
0174
                               System Calls are treated as Breakpoints, Program Exits or I/O
0174
                               Requests. An identifier of '00' is a Software Breakpoint, '01'
0174
                               transfers to the console output handler and '02' transfers to
0174
                               console output.
0174
0174
                               All other ID's are treated as program exits and the identifier
0174
                               is output as part of the exit message.
0174
0174
       5C09 000E*0000
                       LSYSCALL:
                                       LDM
                                              LSVAREA, RO, 15;
                                                                      % Save user regs 0-14
017A
                                      LDCTL
      7D07
                                              RO, NSPOFF;
                                                                      % Save user R15
                                              LSVAREA(30),R0;
017C
      6F00*001E
                                      LD
0180
      21F7
                                              HEX, STACKP ;
                                                                      % Load identifier
0182
0182
                               This routine checks for a valid Breakpoint on a System Call '0'
0182
0182
                                      IF
                                              HEXLOW EO 0
0182
      84FF EEOA
                                      THEN
                                              BEGIN
0186
      31F0 0004
                                              LD
                                                      RO, STACKP (4); % Pick up PC from stack
018A
      6107*0002
                                                      HEX, LBPSAV(2); % Get saved address
                                              LD
018E
                                              SUB
                                                      RO, HEX;
                                                                      % Compare address to PC
```

```
F-16
```

```
BO*HEX: Page 14
MACR08000: Version 2.0 9/05/80
                                                  HEX, LBPSAV(2); & Get saved address to Bo Hex.
MACZ M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
0190
                                                  RO EQ 2
                                                               % If addr compares transfer
0190
     OB00 0002 EE02
                                                  JP LBRKP % to Breakpoint routine
                                           THEN
0196
      5E08*0000
                                           END;
019A
019A
                       This routine handles a request for console output (SC 1)
019A
                                          HEXLOW EQ 1
019A
                                   IF
019A
      OAOF 0101 EE03
                                   THEN
                                           BEGIN
01A0
      5F00*0000
                                                                % Output to CRT then return
                                           CALL
                                                  LPUTLINE;
01A4
                                          IRET % to user program
01A4
      7B00
                                           END:
01A6
                            This routine handles a request for console input (SC 2)
01A6
                                           HEXLOW EQ 2 000 is a Software Breakpoint, 101
01A6
                            Eystem THEN
01A6
01A6
      OAOF 0202 EE03
                                           BEGIN
                                           CALL
                                                  LGETLINE; % Input from CRT then return
01AC
      5F00*0000
                                          IRET % to user program END;
01B0
01B0
      7B00
01B2
01B2
01B2
                           Program Exit routine...
01B2
                                           LHEXVERT;
                                                                % Convert identifier
01B2
      5F00*0000
                                    CALI.
                                          LINEBUFF, #0002; % Store length char
01B6
     4D05*0000 0002
                                    LD
01BC
     6F07*0002
                                    LD
                                           LINEBUFF(2), HEX;
                                                              % Store ident in buffer
                                   LD
                                                               % Set R9 to exit message
01C0
     2109*0000
                                           INDEX, LEXITMSG;
      5F00*0000
01C4
                                   CALL
                                           LPUTLINE;
                                                                % Output message
                                          INDEX, LINEBUFF+1; % Set R9 to output
01C8
     2109*0001
                                   LD
                                           LPUTLINE; % Output identifier
LRESTORE: % Return to monitor
01CC
     5F00*0000
                                    CALL
01D0 5E08*01EC
                                    JP
01D4
01D4 TOR3 EXECUTIVE MODULE
                                   EJECT;
```

```
MONITOR3 EXECUTIVE MODULE
    01D4
   01D4
                                    Break Switch service routine
                                                                      (NMI)
   01D4
   01D4
           5C09 000E*0000
                            LBREAK:
                                                                              % Save user regs 0-14
                                             LDM
                                                     LSVAREA, RO, 15;
   OlDA
           7D07
                                             LDCTL
                                                     RO, NSPOFF;
                                                                              % Save user R15
   01DC
           6F00*001E
                                             LD
                                                     LSVAREA(30), RO;
   OlEO
           2109*0000
                                             LD
                                                     INDEX, LBRKMSG;
                                                                              % Output break message
   01E4
           5F00*0000
                                             CALL
                                                     LPUTLINE:
   01E8
                                             JP
           5E08*01EC
                                                     LRESTORE:
                                                                              % Return to monitor
   Olec
   Olec
   Olec
              0000 EXTERNAL
                                    This common routine saves the user program status from the
   Olec
                                    system stack into the program status save area for all the
              0000 EXTERNAL
   OIEC
                                    service routines. The routine also cleans up the stack pointer
   OIEC
F-17
                                    to prevent stack overflow from interrupts or traps.
   OIEC
                    EXLERNY LRESTORE:
           35F0 0002
   Olec
                                                     RRO, STACKP^(2);
                                             LDL
                                                                              % Pick up program ststus
   01F0
          5D00*0000
                                             LDL
                                                                                    and save it
                                                     LSAVPS, RRO;
   01F4
          A9F5
                                             INC
                                                     STACKP, 6;
                                                                              % Restore stack pointer
   01F6
          5E08*009E
                                             JP
                                                     LCCP;
                                                                              % Exit to monitor CCP
   01FA
   01FA
                                             END.
```

Page 15

MACR08000:

Version 2.0

MACZ M3EXEC O L=B:M3EXEC.PRN W D M P

9/05/80

Page 16 MACR08000: Version 2.0 9/05/80 MACZ M3EXEC O L=B:M3EXEC.PRN W D M P MONITOR3 EXECUTIVE MODULE ASSIGNED LABELS: LALTER 0000 EXTERNAL LBPSAV 0000 EXTERNAL LBPSET 0000 EXTERNAL LBREAK 01D4 LOCAL CODE LBRKMSG 0000 EXTERNAL LBRKP 0000 EXTERNAL LCCP 009E GLOBAL CODE LCMDTBL 0000 EXTERNAL LDELAY 0056 LOCAL CODE LDUMP 0000 EXTERNAL LEXITMSG 0000 EXTERNAL LFLAG 0000 EXTERNAL LGETLINE 0000 EXTERNAL LGO 0000 EXTERNAL LHELP 0000 EXTERNAL LHEXVERT 0000 EXTERNAL LINEBUFF 0000 EXTERNAL 0000 EXTERNAL LINITMSG LMFILL 0000 EXTERNAL LMONIN 00A6 LOCAL CODE LNPSA 0000 LOCAL NPSA LOAD 0000 EXTERNAL LOPCODE 0144 LOCAL CODE LOPMSG 0000 EXTERNAL LPC 0000 EXTERNAL LPRIV 015C LOCAL LPRMSG 0000 EXTERNAL LPROMPT 0000 EXTERNAL LPUTLINE 0000 EXTERNAL LREG 0000 EXTERNAL LRESTORE Olec Local

MACRO8000: Version 2.0 9/05/80
MACZ M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

Page 17

LSAVPS	0000	EXTERNAL	
LSSP	0000	EXTERNAL	
LSTACK	0000	EXTERNAL	
LSTART	0000	GLOBAL	CODE
LSVAREA	0000	EXTERNAL	
LSYSCALL	0174	LOCAL	CODE
LWHAT	0138	GLOBAL	CODE
LWHATMSG	0000	EXTERNAL	

APPENDIX G ASCII CHARACTER SET

Hex	Dec	Char	Hex	Dec	Char	<u>Hex</u>	Dec	Char	Hex	Dec	Char
00	0	NUL	20	32	SP	40	64	e	60	96	
01	1	SOH	21	33	1	41	65	A	61	97	a
02	2	STX	22	34	11	42	66	В	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	С
04	4	EOT	24	36	\$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39		47	71	G	67	103	g
08	8	BS	28	40	(48	72	Н	68	104	h
09	9	HT	29	41)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
OB	11	VT	2B	43	+	4B	75	K	6B	107	k
OC	12	FF	2C	44	,	4C	76	L	6C	108	1
OD	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46		4E	78	N	6E	110	n
OF	15	SI	2F	47	1	4F	79	0	6F	111	0
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	У
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[7B	123	{
1C	28	FS	3C	60	<	5C	92	1	7C	124	1
1D	29	GS	3D	61	=	5D	93]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95		7F	127	DEL



APPENDIX G ASCII CHARACTER SET

	Kell								
					32				
						21			
	64								
							EOT		
							YCK		
						27			
MOI									
		1							
								15	
			0.8	50					
	73								
		V						22	
611		M						23	
					38				
					18				



APPENDIX H Powers of 2 and 16

2n		n
	256	8
	512	9
1	024	10
2	048	11
4	096	12
8	192	13
16	384	14
32	768	15
65	536	16
131	072	17
262	144	18
524	288	19
1 048	576	20
2 097	152	21
4 194	304	22
8 388	608	23
16 777	216	24
Powe	rs of	2

20	=	160
24	=	161
28	=	162
212	=	163
216	=	164
220	=	165
224	=	166
228	=	167
232	=	168
236	=	169
240	=	1610
244	=	1611
248	=	1612
252	=	1613
256	=	1614
260	=	1615

	16n		n
		1	0
		16	1
		256	2
		4 096	3
	(55 536	4
	1 04	18 576	5
	16 77	77 216	6
	268 43	35 456	7
	4 294 96	67 296	8
	68 719 47	76 736	9
1	099 511 62	27 776	10
17	592 186 04	14 416	11
281	474 976 7	10 656	12
4 503	599 627 37	70 496	13
72 057	594 037 92	27 936	14
1 152 921	504 606 84	16 976	15

Powers of 16



	781 = 829	
1 090 511 827 778		
17 592 188 044 418		
72 057 594 037 927 936		



APPENDIX I Hexadecimal and Decimal Integer Conversion Table

	8		7		6		5		4		3		2		1
Hex	Decimal	Hex	Decimal	Hex	Decimal	Hex	Decimal	Hex	Decimal	Hex	Decimal	Hex	Decimal	Hex	Decima
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	268,435,456	1	16,777,216	1	1,048,576	1	65,536	1	4,096	1	256	1	16	- 1	1
2	536,870,912	2	33,554,432	2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	805,306,368	3	50,331,648	3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	1,073,741,824	4	67,108,864	4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	1,342,177,280	5	83,886,080	5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	1,610,612,736	6	100,663,296	6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	1,879,048,192	7	117,440,512	7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	2,147,483,648	8	134,217,728	8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	2,415,919,104	9	150,994,944	9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
Α	2,684,354,560	Α	167,772,160	Α	10,485,760	Α	655,360	Α	40,960	Α	2,560	Α	160	Α	10
В	2,952,790,016	В	184,549,376	В	11,534,336	В	720,896	В	45,056	В	2,816	В	176	В	11
С	3,221,225,472	С	201,326,592	С	12,582,912	С	786,432	С	49,152	С	3,072	С	192	С	12
D	3,489,660,928	D	218,103,808	D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	3,758,096,384	E	234,881,024	E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	4,026,531,840	F	251,658,240	F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
	8		7		6		5		4		3		2		1

To Convert Hexadecimal to Decimal

- Locate the column of decimal numbers corresponding to the left-most digit or letter of the hexadecimal: select from this column and record the number that corresponds to the position of the hexadecimal digit or letter.
- 2. Repeat step 1 for the units (second from the left) position.
- 3. Repeat step 1 for the units (third from the left) position.
- Add the numbers selected from the table to form the decimal number.

To convert integer numbers greater than the capacity of the table, use the techniques below:

Hexadecimal to Decimal

Successive cumulative multiplication from left to right, adding units position.

Example: $D34_{16} = 3380_{10}$

3380

D = 13 x 16	Conversio Hexadecimal	
208		D34
3 = + 3	1. D	3328
211	2. 3	48
x 16	3. 4	4
3376	4. Decimal	3380
4 = + 4		

To Convert Decimal to Hexadecimal

- (a) Select from the table the highest decimal number that is equal to or less than the number to be converted.
 - (b) Record the hexadecimal of the column containing the selected number.
 - (c) Subtract the selected decimal from the number to be converted.
- Using the remainder from step 1(c) repeat all of step 1 to develop the second position of the hexadecimal (and a remainder).
- 3. Using the remainder from step 2 repeat all of step 1 to develop the units position of the hexadecimal.
- 4. Combine terms to form the hexadecimal number.

Decimal to Hexadecimal

Divide and collect the remainder in reverse order.

Exa	ample:	$3380_{10} = D34_{16}$
16	3380	remainder
16	211	4 1
16	13	3
		D

Example:



APPENDIX I rexadecimal and Decimal Integer Conversion Table

				zeił				
		0						
	7							
					A			
			57,344					

To Convert Heradecimal to Declina

- Locale the column of decimal numbers corresponding to the left-most digit or letter of the haxadecimal; select from this column and record the number that corresponds to the position; of the hexadecimal digit or letter.
 - 9 Repeat step 1 for the units (second from the left) position
 - 3. Repeat step 1 for the units (third from the tern) position
- Add the numbers selected from the table to form the declinal number.

To convert integer numbers greater than the dagacity of the table,

lexadecimal to Decimal

Successive cumulative multiplication from left to ngtht, adding units position.

Exemple: D34:6 = 3380:0

To Convert Decimal to Hexadecimal

- (a) Select from the table the highest decirval number trial is equal to or less than the number to be converted.
- (b) Record the hexadecimal of the column containing the selected number.
- (c) Subtract the selected decimal from the number to be
- 2. Using the remainder from step 1(c) repeat all of stap 1 to develop the second position of the hexadecimal (and a ferminant).

 The provided the remainder of the hexadecimal (and a ferminant).
- Using the remainder from step 2 repeat all of step 1 to carvelop the units position of the nexadecimal.
 - A Combine terms to form the hexadecimal number.

ismissbaxeFl of lemissi

"sello saleve) in rebrittiner eat toelloo bins abivio

Example:		
	1 1	

